JU9402V1

Level 0 Data Issues for the ECS Project

White Paper

Working paper - Not intended for formal review or Government approval.

September 1994

Prepared Under Contract NAS5-60000

RESPONSIBLE ENGINEER

Corey Boettcher /s/	9/22/94
Corey Boettcher, PGS Tool Developer EOSDIS Core System Project	Date

SUBMITTED BY

Larry Klein /s/	9/22/94
Larry Klein, Toolkit Manager EOSDIS Core System Project	Date

Hughes Applied Information Systems
Landover, Maryland

Preface

This document contains a summary and an analysis of Earth Observing System (EOS) Level 0 instrument data knowledge and issues. Level 0 data consists of instrument data packets, platform and instrument engineering data and general housekeeping data. Analysis of formats, contents and sources is necessary for the design and implementation of access tools for Level 1 data production processes. These tools are a part of the Science Data Production (SDP) Toolkit and will be delivered with the Toolkit in March, 1995.

In this document (Section 7 and Appendix A) we have proposed a design and implementation of Level 0 access tools. This is a re-design of the tools in the May, 94 Toolkit Users Guide. We have also provided suggestions and examples of usage of these tools. A list of outstanding issues is presented in Section 9.

A principle goal of this document is to solicit feedback from EOS instrument team software developers and scientists, so that software design and calling sequences be placed on firm ground by November this year. Review by the user community is desired and comment welcomed.

Requests for information and comments should be directed to:

Corey Boettcher, cboettch@eos.hitc.com

or

Larry Klein, larry@eos.hitc.com

This page intentionally left blank.

Contents

4. Level 0 Data Formats

Figures

1. Introduction

1.1 Purpose

The purpose of this White Paper is to present an overview of information regarding EOSDIS Core System (ECS) Level 0 data processing. The specific impetus for this work is the need to design Science Data Processing (SDP; formerly Product Generation System (PGS)) Toolkit functions that will provide access to Level 0 data for instrument data production processes. In the course of research done to enable the design of these functions, much useful information has been uncovered. The aim of this paper is to share this material with the greater ECS data processing community, and to encourage discussion about the necessary processing methods, tools and environment for the effective processing of Level 0 data.

Recognition is given to all members of the EOS community who provided the information contained in this document. Special acknowledgment is made to Tom Atwater and Debbie Foch of ECS for their assistance in developing this White Paper.

1.2 Organization

This paper is organized as follows:

Section 1 -- Statement of purpose and overall organization.

Section 2 -- Listing of referenced and applicable documentation.

Section 3 -- Discussion of institutional sources of Level 0 data.

Section 4 -- Overview of known Level 0 data formats, including the structure and content of both Level 0 files and individual packets.

Section 5 -- Availability of orbit and attitude data for Level 0 science data processing.

Section 6 -- Discussion of Level 0 data granules.

Section 7 -- Description of Level 0 access routines to be included in the SDP Toolkit.

Section 8 -- Description of requirements and efforts to create simulated Level 0 data.

Section 9 -- Open issues in regard to use of SDP Toolkit with Level 0 data.

1.3 Review and Approval

This White Paper is an informal document approved at the Office Manager level. It does not require formal Government review or approval; however, it is submitted with the intent that review and comments will be forthcoming.

Questions regarding technical information contained within this Paper should be addressed to the following ECS and/or Goddard Space Flight Center (GSFC) contacts:

- ECS Contacts
  - Corey Boettcher, (301) 925-0751, e-mail: cboettch@eos.hitc.com.
  - Larry Klein, (301) 925-0764, e-mail: larry@eos.hitc.com.
- GSFC Contacts
  - Dan Marinelli, (301) 286-9499, e-mail: dan@marinelli.gsfc.nasa.gov.

Questions concerning distribution or control of this document should be addressed to:

Data Management Office
The ECS Project Office
Hughes Applied Information Systems
1616A McCormick Dr.
Landover, MD 20785

2. Applicable Documents

2.1 Applicable Documents

The following documents are referenced herein and are directly applicable to this document.

Earth Observing System (EOS) Data and Operations System (EDOS) Functional and Performance Specification, 560-EDOS-0202.0004, November 23, 1992.

Earth Observing System (EOS) Data and Operations System (EDOS) Data Format Requirements Document (DFRD), 560-EDOS-0230.0001, December 18, 1992.

Interface Control Document between the Sensor Data Processing Facility (SDPF) and <Mission>, 560-1ICD/0393, August, 1993.

General Instrument Interface Specification, EOS-AM Project, GSFC-420-03-02, December 1, 1992.

Interface Control Document (ICD) Data Format Control Book for EOS-AM Spacecraft (ICD-106), Martin Marietta IS20008658A, April 19, 1994.

General Interface Requirements Document (GIRD) for EOS Common Spacecraft/Instruments, EOS PM Project, Revision A, GSFC 422-11-12-01, January 1994.

Overview of Interfaces Between the Flight Dynamics Facility (FDF) and the EOSDIS Core System (ECS), 554-FDD-91/084 CSC/TM-91/6068, June 1991.

Flight Dynamics Division (FDD) Generic Data Product Formats Interface Control Document, 533-FDD-91/028, June 1991.

Tropical Rainfall Measuring Mission (TRMM) System Specification Space Segment, Revision A, TRMM-490-002, July 16, 1993.

Tropical Rainfall Measuring Mission (TRMM) Telemetry and Command Handbook, TRMM-490-137, February 21, 1994.

Tropical Rainfall Measuring Mission (TRMM) Observatory to Space Flight Tracking and Data Network (STDN) Radio Frequency (RF) Interface Control Document, TRMM-490-086, April 15, 1993.

Detailed Mission Requirements (DMR) for the Tropical Rainfall Measuring Mission (TRMM), Issue 2, 513-4DRD/0193 TRMM-500-135, July 1993.

TRMM Spacecraft Housekeeping Telemetry Packets, Bruce Love, TRMM Project, May 11, 1994.

EOS AM-1 Detailed Mission Requirements (DMR), July 11, 1994.

PGS Toolkit User's Guide for the ECS Project, Version 1, Final, 194-809-SD4-001, May 1994.

Tropical Rainfall Measuring Mission (TRMM) Test and Training Simulator (TTTS) Functional Requirements Document (FRD), 515-4FRD/0194, TRMM-500-154, February 1994.

Level 0 Simulation Discussion presentation vugraphs, Dan Marinelli, DPFT V meeting, April 7, 1994.

Interface Control Document between the Sensor Data Processing Facility (SDPF) and the X-Ray Timing Explorer (XTE) Customers, 560-1ICD/0993, November, 1993.

Interface Requirements Document Between EOSDIS Core System (ECS) and Tropical Rainfall Measuring Mission (TRMM) Ground System, 194-219-SE1-018, June 1994.

Tropical Rainfall Measuring Mission (TRMM) Spacecraft to Clouds and the Earth's Radiant Energy System (CERES) Instrument ICD, TRMM-490-021, April 30, 1993.

Tropical Rainfall Measuring Mission (TRMM) Spacecraft to Lightning Imaging Sensor (LIS) Instrument ICD, TRMM-490-022, February 26, 1993.

CERES Instrument Operations Overview presentation vugraphs, EOS-AM Operations Workshop, Larry Blumfield, April 22, 1993.

Draft CERES Data Catalog Product List, CERES Instrument Team, September 10, 1993.

EOSDIS Test System (ETS) Operations Concept, System Requirements Review (SRR), Version 2, 515-ETS-01-V2, December 1993.

ECS Functional and Performance Requirements Specification, June 2, 1994.

PGS Toolkit Requirement Specification, 193-801-SD4-001, GSFC-423-16-02, October 1993.

2.2 Reference Documents

The following documents are listed for the convenience of the user.

CCSDS Recommendations for Time Code Formats, CCSDS 301.0-B-2, Issue 2, April 1990.

Interface Requirements Document Between EOSDIS Core System (ECS) and NASA Institutional Support Systems, 194-219-SE1-020, June 1994.

Interface Requirements Document Between Earth Observing System (EOS) Data and Operations System (EDOS) and the EOS Ground System (EGS) Elements, 560-EDOS-0211.0001, December 18, 1992.

Spacecraft Simulator (SSIM) Requirements Document (SP-991), Preliminary, Martin Marietta 20008690, June 20, 1993.

Tropical Rainfall Measuring Mission (TRMM) Mission Operations Center (MOC) and TRMM Test and Training Simulator (TTS) System Requirements Review (SRR) presentation vugraphs, Computer Sciences Corporation, August 31, 1993.

Tropical Rainfall Measuring Mission (TRMM) Science Data and Information System (TSDIS) Granule Selection System Impact Considerations White Paper, Mary Shugrue, April 28, 1994.

Interface Requirements Document Between EOSDIS Core System (ECS) and EOS Data and Operations System (EDOS), August 1993.

EOS-AM Instrument Data Base, General Electric Astro-Space Division EOS-DN-IA-008, May 1992.

Preliminary EOS-AM Spacecraft MISR Instrument Flight Operations Understanding, Martin Marietta, August 1993.

Packet Processor II Data Capture Facility (Pacor II) System Operations Concept Document, Revision 1, 560-3OCD/0292, June 1992.

Packet Processor II Data Capture Facility (Pacor II) Functional and Performance Requirements Document, Revision 2, 560-8YRD/0292, May 1992.

Tropical Rainfall Measuring Mission (TRMM) Operations Concept, TRMM-490-080, July 1993.

Tropical Rainfall Measuring Mission (TRMM) System Specification Ground Specification, TRMM-490-003, March 1993.

TSDIS System Requirements Review (SRR) Presentation handouts, February 25, 1994.

Interface Requirements Document Between EOSDIS Core System (ECS) and Flight Dynamics Facility (FDF), August 1993.

The Generic Telemetry Simulator (GTSIM) User's Guide, July 1994.

TSDIS Product Volume Estimates, TSDIS-1994-SPC-0016, T. K. Lin, General Sciences Corporation, June 27, 1994.

This page intentionally left blank.

3. Institutional Level 0 Data Sources

3.1 Introduction

Level 0 data from the EOS instruments onboard the EOS-AM, PM, COLOR, AERO, ALT and CHEM platforms will be made available to ECS by the EOS Data and Operations System (EDOS). Level 0 data from the EOS instruments onboard the TRMM platform will be made available to ECS by the Sensor Data Processing Facility (SDPF), located at the NASA GSFC facility. The focus of this section is limited to those format and ECS processing issues related to TRMM, AM and PM platform data.

3.2 SDPF

TRMM is a joint National Aeronautic and Space Administration (NASA)/National Space Development Agency (Japan) (NASDA) mission that will contain the EOS Clouds and the Earth's Radiation Energy System (CERES) and Lightning Imaging Sensor (LIS) instruments. The TRMM spacecraft will also contain the Precipitation Radar (PR), Visible Infrared Scanner (VIRS) and TRMM Microwave Imager (TMI) instruments. Telemetry data for the PR, VIRS and TMI instruments will be received and processed by the TRMM Science Data and Information System (TSDIS)[1]. Figure 3-1 shows the high level flow of CERES and LIS instrument data from the TRMM platform to ECS via SDPF.

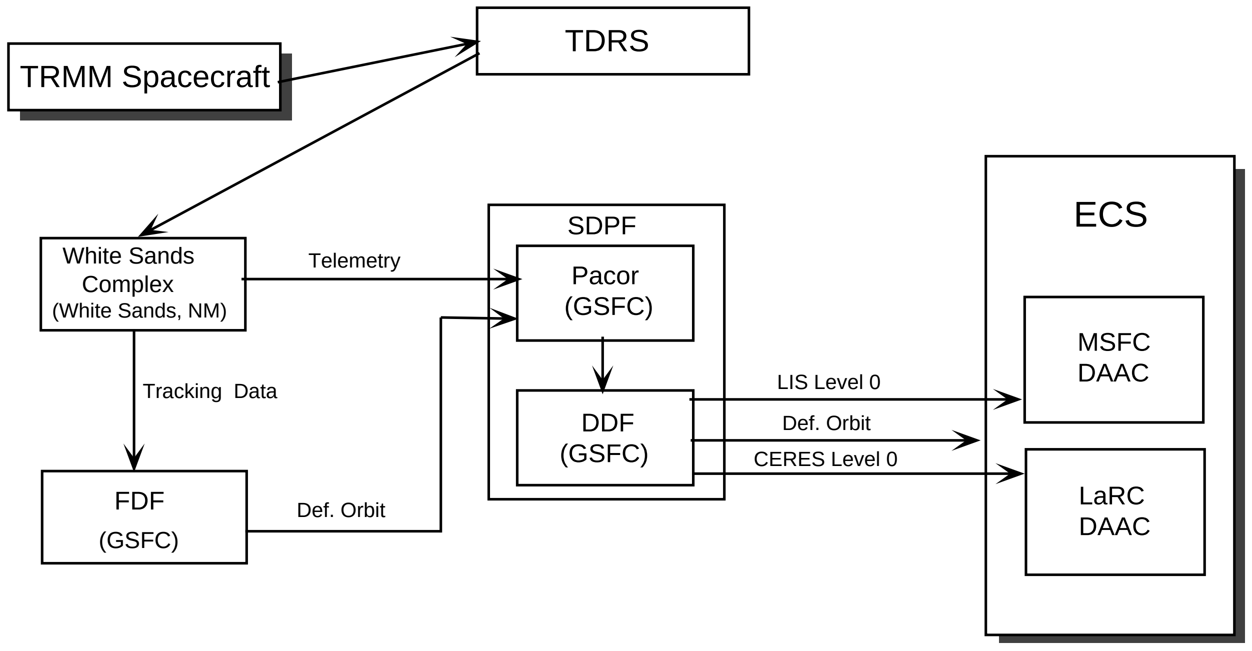


Figure 3-1. TRMM Spacecraft to ECS Level 0 Data Flows

CERES and LIS telemetry data will be relayed by Tracking and Data Relay Satellite (TDRS) from the TRMM platform to White Sands. Tracking data are sent from White Sands to the Flight Dynamics Facility (FDF) at GSFC. FDF generates a daily definitive orbit product and distributes it to the Packet Processing (Pacor) facility, which is part of SDPF. Raw telemetry data are sent from White Sands to Pacor for processing. Once Pacor has processed the telemetry packets into Level 0 data set files and transfers them to the Data Distribution Facility (DDF) within SDPF at GSFC. DDF then sends an electronic Data Availability Notice (DAN) to ECS regarding the Level 0 data[2]. ECS then "pulls" the Level 0 data to the appropriate Distributed Active Archive Center (DAAC) facility. Level 0 CERES data received from DDF will be transferred to the Langley Research Center (LaRC) DAAC for processing by CERES investigators. Level 0 LIS data received from DDF will be transferred to the Marshall Space Flight Center (MSFC) DAAC for processing by LIS investigators[3].

3.2.1 SDPF Data Packaging

SDPF will assemble Level 0 data files that contain instrument data for a 24-hour period and which will be available at DDF once per day[1]. The ECS-TRMM Interface Requirements Document (IRD) lists a requirement for SDPF to process TRMM-platform telemetry data into Level 0 data sets within 24 hours of the last acquisition session.

The basic packaging of Level 0 data by SDPF will be by instrument, which also corresponds to separate Application Process Identifiers (APIDs). SDPF has indicated that it intends to collect packetized telemetry data in three grouping types for transmission to ECS[4]. These three are:

a. CERES instrument science data [APID = 54]

b. LIS instrument science data [APID = 61]

c. A collection of other telemetry packets such platform ancillary and instrument housekeeping and engineering packets grouped in a to be determined (TBD) manner. It is not yet known whether this collection will contain all packetized telemetry data with APIDs other than instrument science data.

The ECS-TRMM IRD[3] contains the requirement that SDPF will retain CERES and LIS Level 0 data sets for five days.

3.2.2 SDPF Level 0 Data Set Format

The basic structure of the Level 0 data sets will be as a collection of Consultative Committee for Space Data Systems (CCSDS) formatted telemetry packets. As received by ECS, Level 0 data will be comprised of header and quality information parameters and a collection of CCSDS-format packetized data[2]. These telemetry data packets may contain instrument science data, platform ancillary data, housekeeping or engineering data.

The SDPF Level 0 data set structure has been well specified by an existing, generic SDPF document (Interface Control Document between the Sensor Data Processing Facility (SDPF) and <Mission>, 560-1ICD/0393, August, 1993)[2]. Although no SDPF documentation yet exists that is specific to the handling of TRMM platform data; a preliminary SDPF-TRMM Interface Control

Document is expected by December 1994[5]. The existing ICD between SDPF and the X-Ray Timing Explorer (XTE) mission[6] currently provides the closest example of the expected SDPF-TRMM ICD and of the SDPF-ECS interface[5].

Level 0 data sets created by SDPF will be comprised of two separate files[2]:

a. a Standard Formatted Data Unit (SFDU) header file and,

b. a Data Set File, consisting of,

    1. Data Set File Header,

    2. Data Set File Source Data Units (i.e., CCSDS-format packets),

    3. Data Set File Quality Accounting Capsule (QAC) and Missing Data Unit List (MDUL).

Current estimates of the size of 24-hour TRMM-platform Level 0 files are 108 MB for CERES and 65 MB for LIS[1,7].

The naming conventions for the SDPF generated Level 0 production data set files are TBD.

3.2.3 SDPF Header Information Formats

The detached SFDU header file will contain parameters such as mission and file IDs, data start and stop times in ASCII format, the time of data set generation. The data set file header will contain parameters such as the spacecraft ID, the number of packets and the spacecraft clock times in PB-5 format of the first and last data packets. Details of the expected type and format of the header parameters within the detached SFDU and the Data Set File can be found in the SDPF-<Mission> ICD[2].

3.2.4 SDPF Quality Information Formats

The Quality Accounting Capsule (QAC) will contain parameters indicating the packet location of the error and a flag specifying the error type. The Missing Data Unit List (MDUL) will indicate the packet sequence counter values of the missing packets. Details of the expected type and format of the QAC and MDUL parameters can be found in the SDPF-<Mission> ICD[2].

3.3 EDOS

Figure 3-2 shows the high level flow of instrument data from the EOS spacecraft platforms to ECS via EDOS.

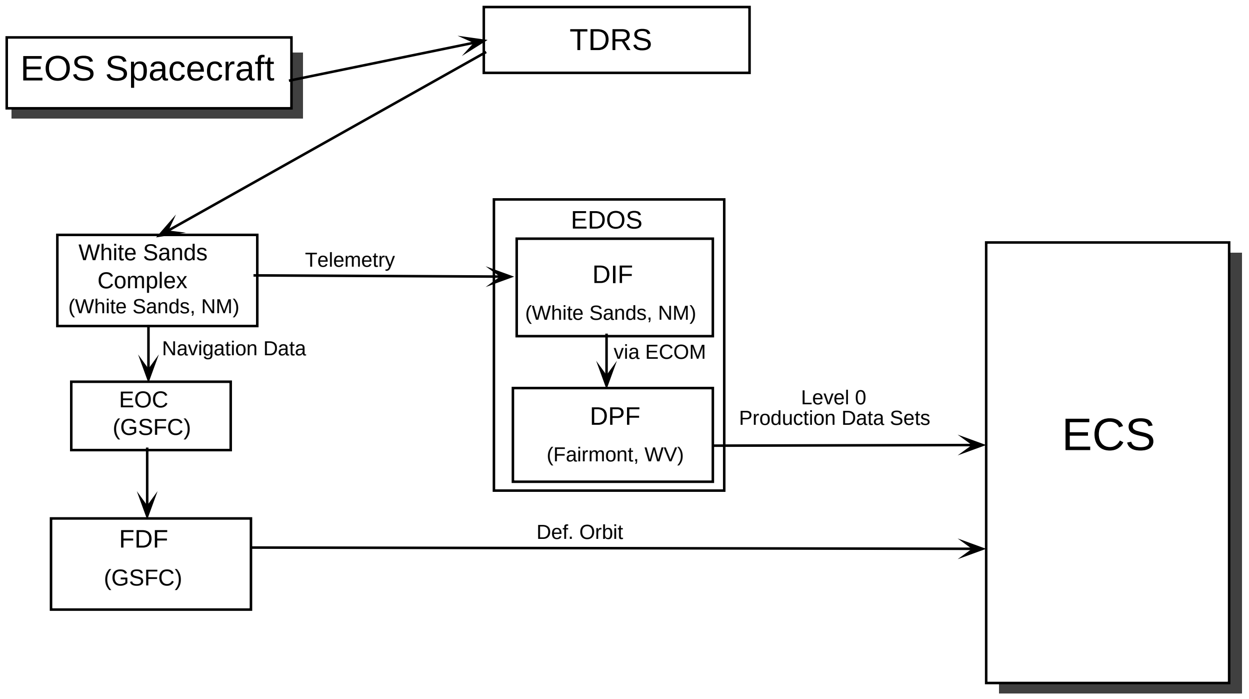


Figure 3-2. EOS Spacecraft to ECS Level 0 Data Flows

As is described in EOS AM-1 Detailed Mission Requirements (DMR) document[8], the EDOS will provide real-time forward and return link data handling services between the White Sands Complex (WSC) and the EOS Operations Center (EOC) to support command and control, and health and safety monitoring functions. Real-time services will be handled at the EDOS Data Interface Facility (DIF) at White Sands, New Mexico.

Level 0 data processing services will be performed once the raw data has been transferred via the EOS Communications (ECOM) network to the EDOS Data Processing Facility (DPF) in Fairmont, West Virginia. This Level 0 processing includes packet time-order sequencing, data transmission artifact removal, data overlap removal and data quality checking. The EOS AM-1 DMR states that the Level 0 throughput capacity of the EDOS DPF is for 24 hours of data at an instrument’s average data rate to be processed within 21 hours of acquisition of the last data packet. The Level 0 data are then transferred to a designated DAAC responsible for further processing. Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Level 0 and housekeeping data are an exception to this and are delivered directly to the ASTER Ground Data System (GDS) via physical media.

EDOS' stated position is that when all the packets needed to produce each data set have been received at the EDOS, the EDOS will send a notification to the ECS in the form of a Data Availability Schedule (DAS). Delivery of the data set from the EDOS to the ECS will be completed within the following 21 hours. Once the raw telemetry has been processed into Level 0 production data sets or quick-look data sets at the DPF, the data sets will be available for ECS to obtain them from EDOS[9].

In January 1994 phone conversation, Gordon Knoble of EDOS stated that a preliminary ICD between EDOS and ECS is expected by December 1994.

3.3.1 EDOS Level 0 Data Packaging

In the minutes of a May 5, 1994, EDOS-ECS Interface Meeting[9], EDOS stated that it will not be granularizing Level 0 data. For the purposes of this White Paper, a "granule" is being defined by ECS as the smallest unit of data required to kick off a Level 1a (or equivalent) instrument data production process (i.e. Level 1a Product Generation Executable (PGE)). Increments of data generated by EDOS may be granules or many increments could make up a granule. For example, processing for a particular instrument may require one orbit of Level 0 data, while increments from EDOS may be smaller.

EDOS' position[9] is that EDOS does not and cannot separate Level 0 data sets into scenes. This would require EDOS to look at the data content, which is not an EDOS function. EDOS has stated that the ways in which EDOS wants Production Data Set (1 VCID/APID set) sizes specified are by[9]:

a. Packet start count to packet stop count

b. Total volume in bytes

c. Wall clock start and stop times

d. Packet start and stop times

EDOS also stated that further data groupings may be made by EDOS on basis of TDRSS sessions, data time ranges, clock time ranges and packet counter ranges[9].

The NASA Advanced Spaceborne Thermal Emission and Reflection Radiometer (ASTER) Instrument Manager raised an issue at the 5/5/94 Interface Meeting as to whether EDOS can keep the ASTER packets in a single virtual channel within one data set regardless of the number of APIDs encountered in that virtual channel. The response from EDOS was, "No." The position of EDOS is that all packets in each EDOS Level 0 data set have the same APID[9].

3.3.2 EDOS Level 0 Data Set Format

The basic structure of the Level 0 data sets will be as a collection of CCSDS formatted telemetry packets. As received by ECS, Level 0 data will be comprised of header and quality information parameters and a collection of CCSDS-format packetized data. Once a Level 0 data set are constructed by EDOS, the data is transmitted to ECS as a Production Data Set (PDS)[8]. Until further information is received from EDOS, the current ECS assumption is that all Level 0 header and quality parameters will be contained in the same physical file as the Level 0 telemetry packets. These telemetry data packets may contain instrument science data, platform ancillary data, housekeeping or engineering data.

The structure and content of Level 0 data sets generated by EDOS are not well defined. ECS currently only has various EDOS requirement documents as references. These documents include EDOS Functional and Performance Specification[10] and the EDOS Data Format Requirements Document (DFRD)[11].

The naming conventions for the EDOS generated Level 0 production data set files are TBD.

3.3.3 EDOS Header Information Formats

The EDOS Functional and Performance Specification (F&PS) includes the requirement that a PDS Construction Record be included with each EDOS-generated PDS. The EDOS F&PS also contains the requirement that each PDS Construction Record will, at a minimum, include the following header information:

a. List of scheduled TSS start times for the PDS.

b. List of scheduled TSS stop times for the PDS.

c. Time of PDS completion.

d. Count of packets in PDS.

e. PDS size (in bytes).

f. Packet sequence count of first packet in PDS.

g. Packet sequence count of last packet in PDS.

Further details on the content and structure of the PDS Construction Record are expected from EDOS by Fall 1994 <To Be Confirmed (TBC)>.

3.3.4 EDOS Quality Information Formats

The EDOS F&PS contains the requirement that each PDS Construction Record will, at a minimum, include the following quality information:

a. List of missing packet sequence counts for the PDS.

b. Count of packets from Virtual Channel Data Units (VCDUs) with errors corrected by Reed-Solomon decoding.

c. List of packets containing EDOS generated fill data including location of first fill octet (i.e. byte) for each packet.

d. Count of packets containing EDOS generated fill data.

e. Count of packets with discrepancies between packet header length code and actual packet length.

f. Count of packets with sequence count discontinuities.

Further details on the content and structure of the PDS Construction Record are expected from EDOS by Fall 1994 <TBC>.

4. Level 0 Data Formats

4.1 Introduction

The purpose of this section is to spell out specific details of Level 0 formats and their corresponding mechanisms, as are currently known. Areas where further details are urgently needed by ECS for continued development and testing of the SDP Toolkit Level 0 access functions are highlighted within this section.

While the information presented here is correct to the best of our knowledge; this White Paper is not intended to serve as a definitive source of Level 0 formats. See section 2 for definitive references.

4.2 CCSDS Packet Data

4.2.1 Telemetry Packet Formats

Telemetry packets within Level 0 data files will be comprised of a primary header, a secondary header and a variable length field containing application data. The primary and secondary headers contain information such as spacecraft clock time, packet sequence number and observing mode that enables the application data to be uniquely identified. The application data field may contain instrument science data, platform ancillary data, housekeeping or engineering data. The platform-specific formats of the telemetry packets are discussed below.

Figure 4-1 shows the format of TRMM platform telemetry packets[1,12,13].

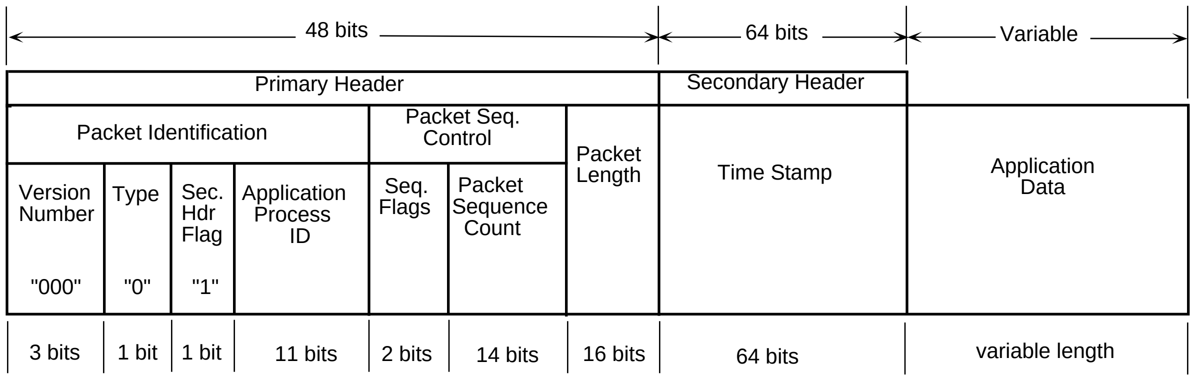


Figure 4-1. TRMM-Platform Telemetry Packet Format

Figure 4-2 shows the format of AM platform telemetry packets[8,14,15].

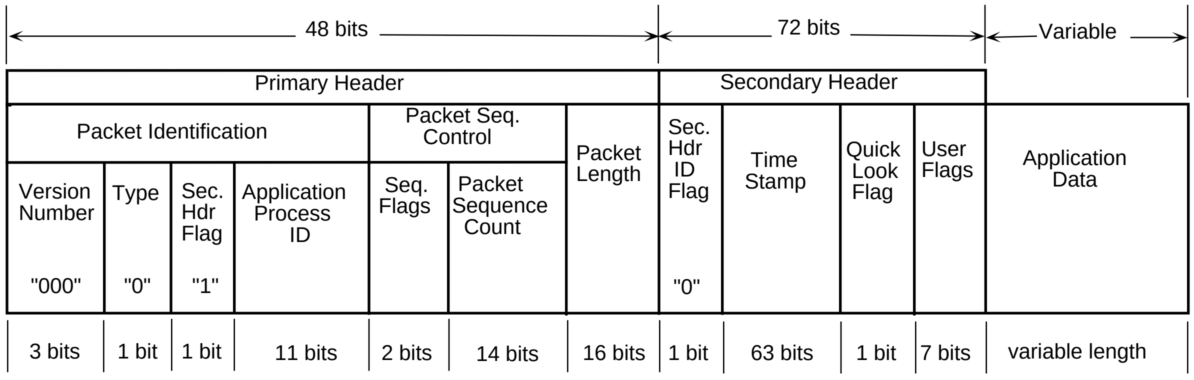


Figure 4-2. AM-Platform Telemetry Packet Format

Figure 4-3 shows the format of PM platform telemetry packets[16].

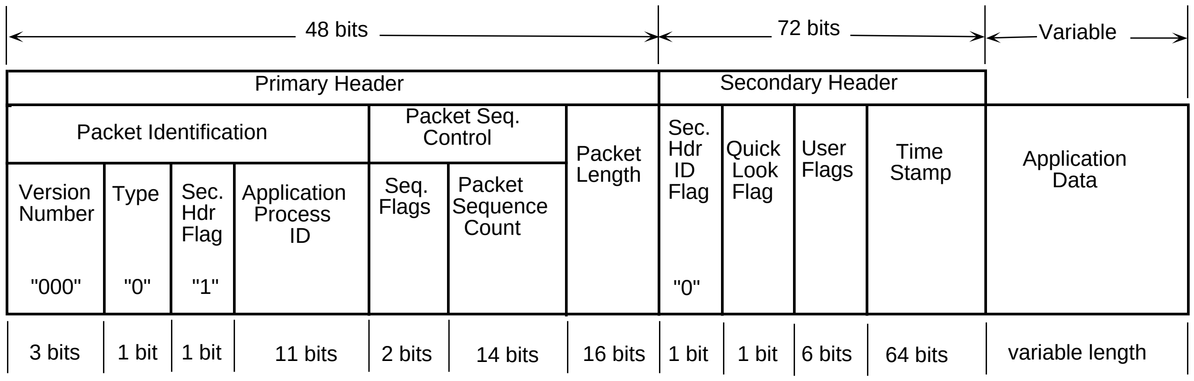


Figure 4-3. PM-Platform Telemetry Packet Format

4.2.2 Packet Identification Fields

The first three fields of the telemetry packet primary headers are set to constant values for AM[14], PM[16] and TRMM[13] platform data. The 3-bit Version Number field will be set to "000". The 1-bit Type field distinguishes telecommand packets from telemetry packets and will be set to "0". The 1-bit Secondary Header Flag field indicates that there is a secondary header and will be set to "1".

4.2.3 Application Process Identifier (APID)

The Application Process Identifier (APID) is an 11-bit field containing a value between 1 and 255 that is used as the primary method for identifying telemetry packets. Each platform assigns APIDs for its housekeeping telemetry, health and safety telemetry, diagnostic telemetry and platform ancillary data, and instrument science data telemetry packets[13,14,16].

On the TRMM platform, for example, CERES housekeeping, science, calibration and diagnostic data packets will have APIDs of 49, 54, 55 and 56 respectively[17]. Correspondingly, LIS housekeeping and science data packets will have APIDs of 50 and 61 respectively[17]. The TRMM Spacecraft Housekeeping Telemetry Packets[17] document contains a detailed listing of the more than 201 APIDs that have been identified for the TRMM mission.

4.2.4 Sequence Flags

For all TRMM[13] and PM[16] telemetry packets, this 2-bit field is always set to "11", since no segmented packets are transmitted to the ground from the TRMM and PM platforms. The document GSFC 420-03-02[15] states the 2-bit sequence flags field in AM platform telemetry packets has been reserved for use by the instruments.

4.2.5 Packet Sequence Count

The packet sequence count is a monotonically increasing 14-bit field that returns to zero upon exceeding the maximum value of 16,383. The packet sequence count of the first packet following a spacecraft power-up will be zero.

The TRMM Telemetry and Command Handbook[13] states that there may be instances, such as instrument restarts or mode changes, which will cause the packet sequence count to be reset to zero in TRMM telemetry packets.

4.2.6 Packet Length

The packet length field in each telemetry packet's primary header contains the length of the entire packet, in bytes, less the length of the primary header (6 bytes) less one byte. Telemetry packets vary in length according to the APID corresponding to the application data field content.

For AM platform data, the lengths of telemetry packets have been specified in ICD-106[14] as ranging from 64 to 7680 bytes for Low-Rate science data and from 128 to 1024 byes for High-Rate science data. However, the minimum length of telemetry packets for Low-Rate science data has also been specified in GSFC 420-03-02[15] to be 128 bytes, not 64 bytes.

For PM platform data, the total size of a single packet associated with an APID will not exceed 8,198 bytes[16].

4.2.7 Secondary Header Identifier Flag

The secondary header identifier (ID) flag is a 1-bit value that is the first bit of the secondary header and which indicates whether this is a non-CCSDS defined secondary header.

TRMM telemetry packets do not contain a secondary header ID flag[1,13]. All 64-bits of TRMM telemetry packet secondary headers are used by the TRMM time stamp.

Although the reference documentation[14,15] is not entirely consistent, it is believed that the first bit of the 64-bit time code in AM platform telemetry packets is actually the 1-bit secondary header ID flag. For AM platform telemetry, this flag will be set to "0" to indicate that the secondary header is a non-CCSDS defined secondary header.

In PM platform telemetry packets, the secondary header ID flag will also be set to "0" to indicate that the secondary header is a non-CCSDS defined secondary header[16].

4.2.8 Time Stamps

The time stamps found in each telemetry packet's secondary header will be in CCSDS format. However, the AM, TRMM and PM spacecraft platforms will each be using a different CCSDS time code format for its spacecraft clock time stamps. The following three sections detail the platform-specific formats of the time stamps.

4.2.8.1 TRMM Packet Data Time Stamps

Time stamps within the secondary headers of TRMM platform data packets will be in CCSDS Unsegmented Time Code Constant and Unit Conversions (CUC) format, with an implicit P-field. The time stamp will be a 64-bit value in the form described in the documents TRMM Observatory to STDN RF ICD[18] and TRMM Telemetry and Command Handbook[13]. The TRMM requirement for maintaining time is that the on-board spacecraft clock be accurate to 1 millisecond[1,19,20]. The spacecraft clock in the time stamp represents the accumulated time in seconds or Mission Elapsed Time (MET) since the power-up of the clock card in the TRMM Spacecraft Data System[13].

The T-field of the TRMM time code is comprised of two 32-bit fields, one for seconds and one for sub-seconds. Figure 4-4 displays the detailed format of the TRMM platform time stamps.

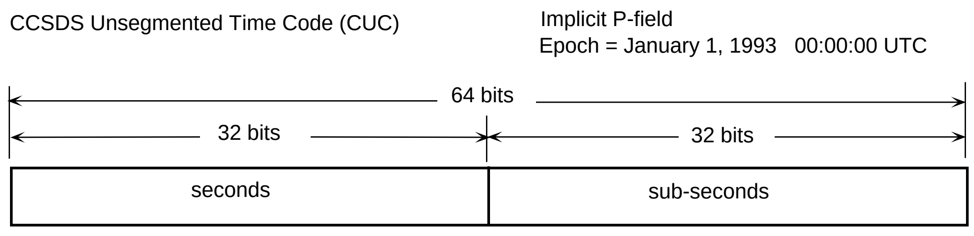

CCSDS Unsegmented Time Code (CUC)
Implicit P-field
Epoch = January 1, 1993 00:00:00 UTC
64 bits
32 bits
32 bits
seconds
sub-seconds

Figure 4-4. TRMM-Platform Data Time Stamp Format

The epoch on which the spacecraft time is based is January 1, 1993, at 00:00:00 Universal Time Coordinate (UTC)[1,13,18], but to obtain this epoch an offset must be added to the MET value in the secondary header time stamp. Previous documentation[12,19,20] stated that this additive factor would be a Universal Time Correlation Factor (UTCF) value located in TRMM platform ancillary telemetry packets. Current information[21] from the TRMM Project indicates that this offset will actually be a parameter located in the SDPF Data Set File Header (see section 4.3.1 for further details). A potential problem arises in that using the UTCF to convert the MET times into UTC under a CUC format, means a continuous time stream with leap seconds removed. This would result in duplicate times within leap second intervals. An alternative would be to use the UTCF to convert to continuous seconds since 1-1-93 UTC noon. This issue has been taken up by ECS with the TRMM project.

4.2.8.2 EOS-AM Packet Data Time Stamps

Time stamps within the secondary headers of AM platform data packets will be in CCSDS Day Segmented (CDS) time code format, with an implicit P-field. The time stamp will be a 63-bit value in the form described in the documents ICD Data Format Control Book for EOS-AM Spacecraft (ICD-106)[14] and General Instrument Interface Specification (GIIS) for the EOS-AM Project[15]. The GIIS states that the AM platform time stamp will have a resolution of 1 microsecond (i.e., the Least Significant Bit (LSB) of the time stamp represents 1 microsecond). FDF has stated[22], however, that the EOS-AM spacecraft clock itself will have a resolution of 10 microseconds and an accuracy of 100 microseconds.

The T-field of the AM time code is comprised of three fields. Figure 4-5 displays the detailed format of the AM platform time stamps.

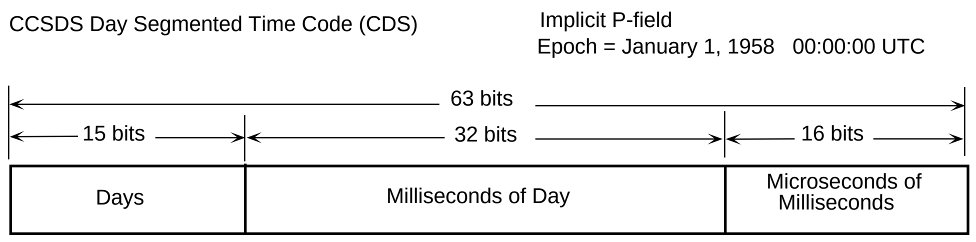

CCSDS Day Segmented Time Code (CDS)
Implicit P-field
Epoch = January 1, 1958   00:00:00 UTC
63 bits
15 bits
32 bits
16 bits
Days
Milliseconds of Day
Microseconds of Milliseconds

Figure 4-5. AM-Platform Data Time Stamp Format

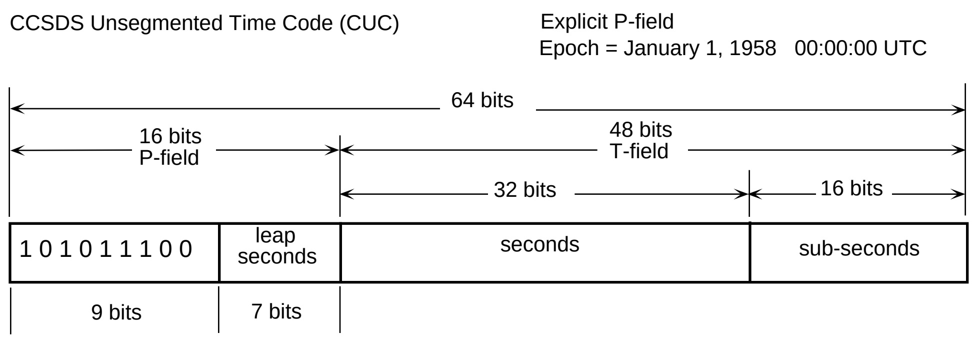
The epoch on which the spacecraft time is based is January 1, 1958, at 00:00:00 (UTC)[14,15]. CCSDS Day Segmented (CDS) time codes are UTC-based, making leap second corrections necessary in recovering any continuous time stream, such as International Atomic Time (TAI), Terrestrial Dynamical Time (TDT) or Barycentric Dynamical Time (TDB). The PGS Toolkit therefore converts the AM Platform times to UTC through the use of the PGS Toolkit routine PGS_TD_SCtime_to_UTC, without need for any leap second correction; note that the resulting seconds field will run up to 60.9999...in a leap second interval. The PGS Toolkit time conversion functions PGS_TD_UTCtoTAI, PGS_TD_UTCtoTDTjed, or PGS_TD_UTCtoTDBjed can then be used to obtain continuous time streams in Julian days or seconds, respectively[23].

It should be noted that there is an unresolved issue regarding the length of the AM platform packet time stamp. The document ICD-106[14] contains references to the time stamp that lists the length of the field as 63-bits in one instance and 64-bits in another. It is believed that the "day" field of the AM platform time stamps is actually 15-bits, not the 16-bits listed; the remaining bit is taken by the secondary header ID flag described in section 4.1.7. As a result, the "day" field of the AM time stamp has a range of 32767, not 65534 days. This discrepancy has been noted by ECS and details have been forwarded to the document author, Martin Marietta AstroSpace Division, the AM-platform contractor.

4.2.8.3 EOS-PM Packet Data Time Stamps

Time stamps within the secondary headers of PM platform data packets will be in CCSDS Unsegmented Time Code (CUC) format, with an explicit P-field[16]. The time stamp will be a 64-bit value in the form described in the document General Interface Requirements Document (GIRD) for EOS Common Spacecraft/Instruments for the EOS PM Project[16]. The GIRD states that PM platform time code data associated with any time mark will be accurate to within 10 milliseconds of the international standard TAI time.

The T-field of the PM platform time code is comprised of one 32-bit field containing the seconds, and one 16-bit field containing the sub-seconds. Figure 4-6 displays the detailed format of the PM platform time stamps.



Details of P-field specification

Bit 0:	"1" denotes P-field will contain a second octet (i.e. 8-bit field)
Bits 1-3:	"010" denote that the time code epoch is Jan. 1, 1958
Bits 4-5:	"11" denote that there are 4 octets of coarse time (i.e. seconds)
Bits 6-7:	"10" denote that there are 2 octets of fine time (i.e. sub-seconds)
Bit 8:	"0" denotes there are no additional P-field continuations
Bits 9-15:	binary number of leap seconds required to specify the difference between UTC time and TAI time

Figure 4-6. PM-Platform Data Time Stamp Format

The epoch on which the spacecraft time is based is January 1, 1958, at 00:00:00 (UTC)[16]. This CUC time code is TAI-, not UTC-based. The explicit P-field contains a 7-bit value of the number of leap seconds required to specify the difference between UTC and TAI, if a conversion is desired. These leap second corrections are expected to be performed during science processing of

Level 0 within the PGS through the use of the PGS Toolkit routine PGS_TD_SCtime_to_UTC, which will automatically apply the leap second correction as the CCSDS packet time stamps are read.

4.2.9 Quick Look and User Flags

Quick-look flags are a single bit in each packet's secondary header. AM[14,15] and PM[16] platform telemetry packets will contain these flags, but TRMM platform telemetry packets will not[1,13]. As previously shown in Figure 4-1, the secondary headers for TRMM telemetry packets will be 64-bits in length and will solely contain the packet time stamp. The CERES science team has raised a concern about the absence of the quick look flag in TRMM platform data, but it's not clear what implications this will have for PGS processing, data handling or staging. For both AM[14,15] and PM[16] platform data, the bit of the quick look flag is set to "1" if the packet is included in a quick look data set, and is set "0" if it is not.

TRMM telemetry packets will not contain user flags in the secondary headers[1,13]. For AM platform data, there are seven 1-bit values reserved in the secondary header as user flags, but ICD-106 states that each of these flags will be set to "0". Telemetry packets for PM platform data have six 1-bit values reserved within each secondary header for user flags[16].

4.2.10 Application Data

The remaining bits in each CCSDS format telemetry packet comprise the variable-length application data field. The length and content of the application data varies based on the source of the data.

The last 16-bits of all application or instrument data fields within EOS-PM platform telemetry packets are reserved for an arithmetic checksum of the data[16]. The information is placed here as a data error indicator to allow determination of where in the data transfer process any errors have occurred.

As stated in the Draft CERES Data Catalog Product List[24] and in vugraphs from a CERES Instrument Operations Overview presentation[25], CERES has reserved the last 64-bytes (512 bits) of each application data field within EOS-platform instrument science packets for duplications of the contents of the EOS-AM platform ancillary packet (described in section 4.3). CERES had arranged for the duplication of the platform ancillary packet information to ensure access to onboard orbit and attitude data. Bill Weaver of the CERES processing team has stated[26] that the CERES instrument team was under the impression that onboard orbit and attitude data would not be separately downlinked and might not otherwise be made available to CERES science processing. Bill Weaver has also stated that although EOS-AM platform CERES science telemetry packets will contain duplicates of the EOS-AM platform ancillary packets, CERES is expecting to access onboard orbit and attitude data from the actual EOS-AM platform ancillary packets, which will be downlinked with a unique APID and made available to the CERES processing string within the PGS.

4.3 Platform Ancillary Packets

4.3.1 TRMM Platform Ancillary Packets

The documents TRMM Spacecraft to CERES Instrument ICD[19] [TRMM-490-021; April 30, 1993] and TRMM Spacecraft to LIS Instrument ICD[20] [TRMM-490-022; February 26, 1993] both refer to all TRMM platform ancillary data as being contained in the same telemetry packet that is stored on-board twice every second and which will be downlinked along with the science data packets for use in ground processing. Using the document TRMM System Specification Space Segment[12] [Rev. A; TRMM-490-002; July 16, 1993] as an information source, these Instrument ICDs each provide a table listing the data types to be included in the ancillary data packet. However, a Configuration Change Request (CCR) was made to TRMM-490-002 on December 17, 1993, by Bruce Love of the TRMM Project. The CCR states that the Geodetic Altitude parameter will no longer be included as part of the TRMM platform ancillary data. This CCR also indicates that TRMM observatory ancillary data types listed in TRMM-490-002 (and in the Instrument ICDs) will be not be located in a single TRMM telemetry packet. This CCR states that:

a. It is less of a strain on the TRMM on-board storage and telemetry utilization if this data can be contained in multiple packets, each of which is stored at the appropriate rate for the associated data. This change still provides the required data to the ground for science data processing. There is NO impact to any instrument with this change. The description of "ancillary data" in the Instrument ICDs was included in order to describe what spacecraft data will be available on the ground for science data processing. This data is NOT part of the "spacecraft to instrument" interface.

Bruce Love has also written the document TRMM Spacecraft Housekeeping Telemetry Packets[17] (which will eventually become part of the Appendix A of the TRMM Telemetry and Command Handbook[13] [TRMM-490-137]). This document contains the TRMM telemetry packet types listed according to Application Process Identifier (APID), as well as detailed packet telemetry definitions for each APID. The information in this document indicates that TRMM platform ancillary data referenced in the Instrument ICDs will be stored in the following two telemetry packets:

a. Attitude Control System (ACS) Ancillary Data Packet (APID=45) -- This 70 byte packet will be output by the TRMM ACS once every 0.5 seconds. This packet will contain the TRMM observatory attitude errors, pitch, roll and yaw errors, the ACS spacecraft orientation and the ACS control mode. This packet contains all the data types listed in TRMM-490-002, except for Geodetic Altitude (which has been deleted) and the on-board predicted Position and Velocity vectors (which now appear in the ACS 2 Hertz Mode Telemetry Packet).

b. ACS 2 Hertz Mode Telemetry Packet (APID=17) -- This 526 byte packet will be output by the TRMM Attitude Control System (ACS) once every 4.0 seconds. This packet will contain the orbit data (predicted position and velocity vectors) which are computed on-board the TRMM spacecraft, based on FDF-supplied predicted orbit products.

The fact that the TRMM ancillary data is now in two separate telemetry packets should not impact SDP processing since it is *not* expected that the orbit data in the ACS 2 Hertz Mode Telemetry Packet (APID=17) will routinely be used in the processing of CERES and LIS Level 0 data[21]. The TRMM Project has stated that although the position and velocity vectors calculated on-board the TRMM spacecraft will be downlinked with TRMM telemetry data, these on-board orbit data will not be of sufficient accuracy to be used in scientific processing[27]. As will be described in section 5.4 of this White Paper, a daily Definitive Orbit product generated by FDF will the official source of TRMM orbit data for use in the processing of TRMM-platform CERES and LIS Level 0 data.

Although the Instrument ICDs[19,20] state that the TRMM UTCF will be reported in the ancillary data packet, the current information is that this time correlation factor will not be present in the ACS Ancillary Data Packet (APID=45). According to the TRMM Project[21], a time value found in the SDPF Data Set File Header will be used to correlate the mission elapsed time (MET) found in each telemetry packet's secondary header with a time reference such as UTC. The SDPF Data Set File Header contains a value of the spacecraft clock for the first source data unit (i.e., telemetry packet) within the Level 0 data set file[2]. The SDPF-<Mission> ICD[2] describes this parameter as follows, "A 72-bit field containing the UTC equivalent of the spacecraft clock reading, rounded to the nearest microsecond and encoded to parallel binary format (PB-5) time code, at the beginning of the first source data unit of the data set contained in the file."

The current position of the TRMM Project[21] is that comparing this PB-5 time with the MET found in the first data packet's time stamp provides an offset in seconds that can be used to convert the MET of the time stamps in the other telemetry packets to another time reference such as UTC.

4.3.2 AM Platform Ancillary Packets

The AM platform ancillary packet is a 64-byte CCSDS format packet with its own APID[15]. It contains on-board orbit and attitude parameters from TDRSS Onboard Navigation System (TONS). Ancillary packets are generated by the AM spacecraft every 1.024 seconds and supplied to the instruments[14]. The telemetry packets containing the EOS-AM platform ancillary data are downlinked with the other telemetry data at a TBD rate.

The EOS-AM platform ancillary packet also contains solar and lunar positions[15] that are calculated, based on an input spacecraft clock time and algorithms, by the Astrodynamic Processing segment of the onboard Navigation Software.

Detailed content and format listings of the AM platform ancillary packet are included in the EOS-AM Project document General Instrument Interface Specification (GIIS)[15].

4.3.3 PM Platform Ancillary Packets

No information is currently available as to whether PM-platform telemetry data will contain any platform ancillary data packets.

4.4 Platform Engineering, Housekeeping and Health & Safety Packets

Details of the structure and parameter content of other engineering, house-keeping and health and safety packet data are TBD.

5. Orbit & Attitude Data Accessibility

5.1 Introduction

There will be two sources of orbit and attitude data available during the processing of Level 0 instrument data, platform ancillary data packets[13,15,17] and data from the Flight Dynamics Facility (FDF)[8,29,30].

5.2 On-board Orbit/Attitude Data

Platform ancillary data packets are downlinked in telemetry computed by the spacecraft on-board computer and will contain some orbit and attitude data in TRMM and EOS-AM mission data.

The on-board computer computes its orbit information by propagating vectors derived from orbital elements supplied by FDF daily. For the EOS-AM mission, there is Tracking Data Relay Satellite (TDRS) Onboard Navigational System (TONS) software onboard that computes the orbit by receiving Doppler data from the TDRS satellites[28].

Spacecraft attitude is computed onboard based on attitude sensor measurements.

5.3 Flight Dynamics Facility Orbit/Attitude Products

FDF computes orbit information based on tracking data received from TDRS satellites, or ground stations as backup as needed during real time contacts. This tracking data are in the form of range and range rate (Doppler) information. The tracking data are stored in a database and processed to obtain position and velocity vectors during the interval spanning the tracking data processed, nominally one to two days. This ephemeris covering the tracking data interval is called a definitive orbit. The orbit solution (vectors and elements) at the desired epoch, usually 0 hours UTC within the time interval of the tracking data, is written to the header of ephemeris output file. The epoch solution is then propagated forward for the desired length of time to generate a predicted orbit, which is written to the ephemeris file at 60-second intervals. Ephemeris vectors are propagated based on a model of environmental torques that include solar flux, spacecraft mass and drag profile, and height of the atmosphere.

FDF determines spacecraft attitude using attitude sensor data (star tracker, Sun sensor, Earth sensor, Inertial Reference Unit (IRU)/gyro, magnetometer) included in telemetry received during the real time contacts. This telemetry is relayed from the EOS Operations Center (EOC) to the FDF via the local network as soon as it is received from the TDRS or ground station and is a subset of the total raw telemetry. FDF monitors this telemetry, determining attitudes, checking the status of the attitude sensors, and monitoring any spacecraft orbit or attitude maneuvers. FDF can also use the real time telemetry to do a rough check on the onboard attitude and orbit determination accuracy.

Twice per orbit, during a TSS, raw telemetry from the onboard tape recorders is downlinked at high speed and sent to the Data Capture Facility (DCF). Selected attitude sensor data, orbit and attitude data computed onboard, and engineering data are stripped off and provided to the FDF in a single electronically transmitted file for analysis of the quality of onboard attitude and orbit determination. The onboard attitude is compared with attitudes determined in FDF, which are based on more data and a definitive orbit. Onboard attitudes are determined based on only seconds of sensor observations at a time and use the orbit vectors determined onboard. When the difference in ground and onboard attitudes exceeds requirements the FDF will provide repaired or refined replacement attitudes.

A refined orbit is really a definitive orbit, but covers only the time period when the onboard orbit data was not reliable[31]. It is generated in response to a request from the ECS project, and is provided in the format of definitive ephemerides.

FDF provides a star catalog to the EOC, which is loaded onboard the spacecraft for use by the star trackers in identifying stars for attitude determination and control. FDF provides EOC with planning aids that include maneuver planning information, predicted orbit data in the form of either Extended Precision Vectors (EPVs) for TDRSS and the spacecraft, predicted ground track, predicted TDRSS and ground station contact times, orbit events such as ascending/descending node times and South Atlantic Anomaly (SAA) crossing times, and high gain antenna pointing angles. These planning aids are provided to the EOC via the EOS Communications (Ecom) network.

5.4 Orbit/Attitude Input to ECS Processing

Orbit and attitude data will be used as input to the EOS investigator supplied Level 0 processing software in the PGS. For data processing of TRMM-platform data, the primary ephemeris data sources are FDF-generated orbit data and on-board attitude data[22,27]. For data processing of AM-platform data, the primary ephemeris data sources are on-board orbit data (from TONS) and on-board attitude data[15,31]. For data processing of PM-platform data, the primary ephemeris data sources are TBD, but may be FDF-generated orbit data and on-board attitude data.

Table 5-1 lists the primary and backup sources for orbit and attitude data for the TRMM, AM and PM missions.

Table 5-1. Orbit and Attitude Data Sources

Mission	Orbit Data	Attitude Data
TRMM	P: EPHEM (daily) B: Platform Anc. Pkt (per orbit)	P: Platform Anc. Pkt (per orbit) B: ?
EOS-AM1	P: Platform Anc. Pkt (per orbit, from TONS) B: FDFEPHEM (as needed)	P: Platform Anc. Pkt. (per orbit) B: FDF Definitive Attitude (as needed)
EOS-PM1	P: ? FDFEPHEM (daily) B: ?	P: ? Platform Anc. Pkt (per orbit) B: ? FDF Definitive Attitude (as needed)

P = Primary source, B = Backup source

(*) = frequency of data generation

Platform Ancillary Pkt - ancillary packet downlinked with telemetry data
FDFEPHEM = ASCII format of FDF Definitive Orbit
EPHEM = Binary format of FDF Definitive Orbit

Access of this orbit and attitude data by the science processing software will be through the SDP Toolkit routine PGS_EPH_Ephem_Attit. This toolkit function is designed to provide a uniform interface for accessing orbit and attitude data from a variety of sources. ECS' assumption is that ephemeris data will either be in the form of CCSDS format telemetry (i.e., ancillary) data packets or as FDF-generated products. The circumstances and method of "switching" from a primary to secondary source of ephemeris data within the ECS data production environment is TBD.

The form of TRMM platform on-board ephemeris data will be the ancillary data packet described in section 4.3.1 and the format is defined in TRMM Project documentation[13,17]. The form of AM platform on-board ephemeris data will be the ancillary data packet described in section 4.3.2 and the format is defined in AM Project documentation[15]. The format (and existence) of PM platform ephemeris data is currently unknown. The formats of the FDF definitive orbit products, EPHEM and FDFEPHEM, are defined in the Flight Dynamics Division (FDD) Generic Data Product Formats ICD[30].

The SDP Toolkit[23] includes the program PGS_EPH_attOrbSim, which simulates orbit position and attitude data for EOS spacecraft. The initial version of this program supports the TRMM, EOS-AM and EOS-PM platforms. The program returns spacecraft position information in Earth Centered Inertial (ECI) coordinates, given the inputs of a spacecraft identifier, desired UTC start and stop times and the desired time interval for the calculation of the output orbit position and attitude data. It then calculates the spacecraft reference frame to ECI frame coordinate transformation quaternion from this data. The PGS_EPH_attOrbSim program returns spacecraft position, velocity, roll, pitch and yaw, and roll, pitch and yaw rates. The initial version of this tool will not return portions of days; the output will be in increments of 24 hours, starting at 12AM UTC of the day of the input UTC start time. An input parameter also allows the user to specify that the routine should introduce small random variations (i.e. noise) to the spacecraft yaw, pitch, and roll and/or yaw, pitch, and roll rates.

This page intentionally left blank.

6. Level 0 Data Granules

6.1 Level 0 Data Granules

A major concern is that no group or effort within ECS, EDOS or SDPF is currently scoped to granularize Level 0 data.

As is described in section 3.3.1, EDOS has no plans to granularize EOS-platform Level 0 data based on the science content or Earth location of the instrument science data[9].

SDPF has no plans to granularize TRMM-platform instrument science data[5]. This is likely less of a concern for ECS data processing due to the relatively low volumes for TRMM-platform CERES and LIS Level 0 data. The July 1993 *TRMM Detailed Mission Requirements Document* states that SDPF will transmit TRMM-platform CERES and LIS Level 0 data once daily to ECS.

Joe Guzek of ECS has stated that ECS' position is that granularization of data from EOS spacecraft will be done by either by EDOS or by science team supplied algorithms if EDOS generated PDSs are not acceptable. The Ingest Client service of ECS will archive Level 0 data at the DAACs in the form received from SDPF/Pacor and EDOS. Granularization of Level 0 data will not be performed by ECS prior to archiving, since ECS has a requirement to supply Level 0 data sets back to EDOS if there is a problem with data stored in EDOS archives. As a result, any reprocessing of the Level 0 data will also require regranularization of the data. The added processing impact of regranularization needs to be taken into account when reprocessing is scheduled.

Table 6-1 lists the Level 1 granule requirements for TRMM, AM and PM platform data as specified by the EOS instrument teams. The data granule size requirements were collected by the EOS instrument teams. The data granule size requirements were collected by the EOS Project Science Office (SPSO) as part of a recent NASA/SPSO survey of the EOS instrument teams[32].

Table 6-1. EOS Investigator Granule Size Requirements

Instrument	Platform	Level 1
AIRS	PM	One orbit
AMSU -A	PM	One orbit
ASTER	AM-1	One scene (60 km x 60 km box)
CERES	AM, PM, TRMM	One day
EOSP	AM2	One orbit
LIS	TRMM	One orbit
MHS	PM	One orbit
MIMR	PM	One orbit
MISR	AM	One orbit
MODIS	AM, PM	2330 km x 2330 km box
MOPITT	AM	One day

The current SDP Toolkit assumption is that the above Level 0 granules will be available to Level 1 PGEs in one or more physical files. Each file will contain its own header (metadata) information, including quality flags, number of packets, time range, and so on. Once Level 0 data arrives at a DAAC, and which includes data within the time range requested by a Level 1 PGE, the data will be staged to online disk and the PGE kicked off. The Toolkit will provide access to files in time sequence. Tool functionality will include file header access, file opens and software which will break packets out of the files. We intend to provide code fragments which will set up loops for accessing time ordered files and read packets within the files. Users will modify these fragments to construct Level 1 granules in the manner they choose. For example, processing decisions may be made based on quality flags. As another example, Toolkit routines will return a warning message if a packet is out of the requested range, however, an instrument process may choose to include the packet.

Note: If Level 0 data is granularized to a single file prior to user access, the current design does not hinder that access. The processing loop (described in Appendix A) of file opens for all physical files then becomes a single open.

Details on Level 0 data access via the Toolkit are presented below in Section 7 and in Appendix A.

7. SDP Toolkit Level 0 Access Functions

7.1 Introduction

The purpose of this section is to present the design of SDP Toolkit functions to be used to access Level 0 data by science software.

The SDP Toolkit is the interface between the science software and the ECS data production environment. Its purposes include assuring that this interface is standard across science instruments, enabling portability across computing platforms, and reducing common coding effort by science teams. An additional purpose of the Toolkit is to supply users with software development tools, such as Level 0 data formatters and platform navigation and attitude simulators.

Due to the nature of Level 0 processing, Level 0 access functions must be as fast as possible; every effort is made in this design to assure this.

Note that the design presented here is markedly different from that presented previously in "PGS Toolkit Users Guide for the ECS Project, Version 1 Final, May 1994." The main driver behind the changes is to allow for maximum flexibility of science software processing, particularly as regards quality data handling and dynamic memory allocation. This design is currently in draft form; it is intended to finalize the calling sequence interface of these tools by November, 1994. Comments and suggestions regarding the design are appreciated. Appendix A contains further details on the usage of SDP Toolkit Level 0 access routines, as well as preliminary calling sequences.

7.2 Tool Descriptions

The current design of the SDP Toolkit contains five functions that will provide information about and access to Level 0 data. One function initializes necessary variables for other Level 0 functions and returns metadata regarding the number of physical files and time range of Level 0 data staged to a PGE. One function returns, for a single staged physical file of Level 0 data, the file metadata attributes that are stored in a PDPS Process Control table. Two of the functions open Level 0 data files and return file header and quality data; one returns the SDPF- and the other returns the EDOS-specific header and quality information. The packet read function returns a single CSSDS format telemetry packet. A sixth function, for use in testing, returns a simulated telemetry packet which is of CCSDS format, but contains fill data in place of actual instrument science data. Development is ongoing to enable this tool to be coupled with the functionality of the SDP Toolkit platform ephemeris simulator tool PGS_EPH_attOrbSim described in section 5.4. In Section 7.2.7, we provide a summary of usage of these tools and high level pseudo-code, which shows an example of how the tools would be used in sequence. Calling sequences, input and output parameter and data descriptions, usage examples, and other details about these tools are presented in Appendix A.

The tools below are currently scheduled for delivery as part of the SDP Toolkit in March, 1995. Calling sequences for these tools (Appendix A) will be delivered in the Toolkit Users Guide in November, this year. They will be modified as necessary, based on community feedback to this White Paper.

7.2.1 PGS_IO_L0_Init

This function initializes various global variables necessary for the proper functioning of the Level 0 Open and Read packet tools. This initialization of Level 0 Toolkit functions performs a mapping of the file version number with a time-ordered file version number, which is the identifier the science software utilizes. This lower level mapping of the file version number is necessary since if multiple physical files of Level 0 data are staged, they may not be staged in time order. The Toolkit will insure that files accessed by the user are in time order.

This function also returns information regarding the time range and number of physical files for the Level 0 data that has been staged to a PGE. It is intended that returning select metadata to the science software using PGS_IO_L0_Init will reduce the need for individual calls by the science software to the routine PGS_IO_L0_GetFileAttr for each physical Level 0 data file. PGS_IO_L0_Init must be called prior to using any of the other Level 0 access routines.

7.2.2 PGS_IO_L0_GetFileAttr

This function returns file attributes for a single physical file of Level 0 data. The PGS_IO_L0_GetFileAttr function provides transparent access to time-ordered file attributes for Level 0 data files. The file attributes are a TBD collection of ECS-internal metadata regarding the file content and quality. NOTE: It is expected that these file attributes will be located in a SDPS Process Control table. The exact generation method, parameter number and content, and storage format(s) for these file attributes are TBD.

This function is actually a wrapper on top of the SDP Toolkit PGS_PC_GetFileAttr routine. This wrapper is necessary since the version numbers of the Level 0 data files staged to a PGE may not be in time order. This function internally translates the input file version number to the time-ordered file version number and then retrieves the file attributes for that physical file. It is intended that previously returning select metadata to the science software using PGS_IO_L0_Init will reduce the need for individual calls by the science software to PGS_IO_L0_GetFileAttr for each physical Level 0 data file.

7.2.3 PGS_IO_L0_SDPF_Open

This tool opens a Level 0 production data set generated by SDPF and returns buffers containing the SDPF-specific header and quality information. When supplied with the input of the file logical[23] of the Level 0 data set file, this tool will return the physical file name of the Level 0 data set file and three buffers of metadata: one contains the detached SFDU header, one contains the data set file header and one contains the data set file quality information (Quality Accounting Capsule and Missing Data Unit List entries).

7.2.4 PGS_IO_L0_EDOS_Open

This tool opens a Level 0 production data set generated by EDOS and returns the physical file name and buffers containing the EDOS-specific header and quality information. When supplied with the input of the file logical of the Level 0 data set file, this tool will return one buffer of metadata containing the header and quality information physically located in the Level 0 data set file.

7.2.5 PGS_IO_L0_Read_Pkt

This tool takes as input the file handle corresponding to the logical file and logical file version number of the desired Level 0 data file. Also required as input is an identifier (spacecraftTag) specifying which spacecraft platform this Level 0 data originates from, which is necessary since the bit-level locations of the individual values within the packet secondary header, including the time stamp, are platform dependent. The output of this tool is a single Level 0 CCSDS format packet.

The parameters within the packet primary and secondary header are returned to the science software as separate variables. The time stamp within each packet secondary header is returned in the bit-packed, platform-dependent format. In order to track the packet times, it is necessary for the SDP Toolkit to convert the platform dependent time stamp into a more easily used, internal format. The value of the individual packet time stamp converted to this SDP Toolkit internal format (TAI, as continuous seconds since 12AM UTC 01-01-1993) is also returned to the science software. For any additional conversions of the platform-dependent time stamp, the science software must then make a call to a SDP Toolkit routine such as PGS_TD_SCtime_to_UTC. The data contained within each packet's variable length application data field is returned to the science software as an unformatted data buffer.

7.2.6 PGS_IO_L0_Write_Pkt

As a result of SDP Toolkit development and testing, a tool is being developed which would generate a simulated CCSDS format packet. There will, however, be no attempt to simulate any science content within the telemetry packet. This tool will simulate the CCSDS packet structure of Level 0 telemetry. The tool is a result of efforts to support the ECS development and internal testing of the SDP Toolkit Level 0 access routines. This tool is not designed to be used during routine science processing. The packetized output of this tool will have the structure of a CCSDS format packet, but the variable-length application data field nominally generated will contain fill or zeroed data. Users can insert their own data values into the packets by supplying the routine with an input buffer containing the packet application data field, for purpose of testing data production code. This packetizing tool will be capable of generating a simulated telemetry packet of CCSDS format with variable time stamps, packet sequence counters, APIDs and packet lengths. The individual TRMM, AM and PM mission specific telemetry packet formats described in section 4.2.1 will be supported. This tool will be supplied as part of the SDP Toolkit to help support the development of science software by EOS investigators.

Although a design and calling sequences for this tool have not been finalized, ECS intends for this tool to have the capability of outputting packet time stamps that simulate the TRMM, AM and PM mission specific time code formats. Therefore, the SDP Toolkit time conversion routines used to unpack the time stamps of simulated Level 0 packets will be the same routines described in section 4.2.8 that will be used to unpack time stamps from post-launch Level 0 telemetry packets.

7.2.7 Summary of Tool Usage by Science Software

This section contains a brief summary of how science software might use the above tools to do Level 0 processing.

First, the science software PGE must select which of the staged Level 0 data files it wishes to process. Required Level 0 data is assumed staged by the production system scheduler. Processing steps would be as follows:

a. Using the SDP Toolkit function PGS_IO_L0_Init to determine the number of physical Level 0 data files that are currently staged and what the time range of the staged data is;
b. Then, if more detailed file metadata is required, a processing loop on this number of files is constructed to individually query the attributes (i.e., metadata) of the staged files using the function PGS_IO_L0_GetFileAttr.
c. If this is the desired file, open it using either PGS_IO_L0_SDPF_Open or PGS_IO_L0_EDOS_Open.
d. Read packets until done, using the Toolkit function PGS_IO_L0_Read_Pkt to return single packets. (Memory to store the packets returned must have been previously allocated by the science software.)
e. Repeat for all Level 0 files desired.

Below is a pseudo-code outline of how these Level 0 access routines might be used during science software processing.

```
Call the Level 0 initialization routine to get the number of and time range of the
staged Level 0 data files; this also sets up the global variables necessary for the SDP
Toolkit to provide time-ordered access to the staged Level 0 data files.
call PGS_IO_L0_Init(num_files, time_first, time_last,...)

Loop through num_files physical Level 0 data files to read in the Level 0 data.

       If necessary, read in the ECS-internal file attribute for this Level 0 data
file.
       call PGS_IO_L0_GetFileAttr(fileattr, ...)

       Open a Level 0 data file and get the file handle; this also returns the header
and
       quality metadata located within in the Level 0 data file.
       call PGS_IO_L0_SDPF_Open(sdpf_hdr_1, sdpf_hdr_2, sdpf_qual, file_handle, ...) or
             PGS_IO_L0_EDOS_Open(edos_hdr_qual, file_handle, ...)
```

```
        Loop through the Level 0 data file and read in the packet data.

                call PGS_IO_L0_Read_Pkt(pkt_apid, pkt_seqcount, pkt_length, pkt_time,
                                       pkt_data, ...)

                Store packet header parameters (APID, packet number, packet length, time
stamp,
                etc.) and application data.

        end
end
```

7.2.8 Usage of Level 0 Packetizing Tool

If desired, the function PGS_IO_L0_Write_Pkt can be used to generate simulated Level 0 test data. The current design is for the routine PGS_IO_L0_Write_Pkt to be called by the user within a processing loop in order to generate a number of CCSDS format packets. Packet lengths, APIDs, packet sequence numbers and platform-dependent time stamps are among the planned user-defined inputs to this packet generation tool. Users will be able to insert their own data values into the packets by supplying the routine with an input buffer containing the packet application data field.

Further details of the design and usage of this tool are TBD.

7.3 SDP Toolkit Design Assumptions

7.3.1 SDP Toolkit Responsibilities

The SDP Toolkit Level 0 access tools have the responsibility for:

a. Opening a single Level 0 data file, upon being passed a logical file name and logical file version number.

b. Reading the header and quality data contained in the Level 0 data file into a buffer(s).

c. Reading the primary and secondary header parameters of a single Level 0 CCSDS format packet and returning them to the science software. The Toolkit will return a warning if the packet is outside the requested time range. The values of the individual packet times converted to an SDP Toolkit internal format (TAI, as continuous seconds since 12AM UTC 01-01-1993) are also returned to the science software.

d. Reading the application data field of a single Level 0 CCSDS format packet and returning the data to the science software in the form of a buffer.

e. Insuring that files are in time-ordered sequence if there are more than one within the requested time range.

f. Providing code fragments which will execute the above functionality. These fragments will be modified by users to fit individual needs. It is the intention of the Toolkit to provide the *basic* code necessary to construct Level 0 data granules, short of breaking into the instrument- and mode-specific application data fields themselves. That will be the responsibility of the science software.

7.3.2 Science Software Responsibilities

The science software is responsible for:

a. Using the function PGS_IO_L0_Init to obtain the number of physical files (i.e., number of versions) which have been staged and the time range of the staged data.

b. Using the function PGS_IO_L0_GetFileAttr to retrieve staged attributes (i.e., metadata) about each of these staged files as necessary.

c. Accessing metadata values within the file attribute string returned from the tool PGS_IO_L0_GetFileAttr.

d. Unpacking the Level 0 data returned by the packet read tool.

e. Allocating memory to store the data returned.

7.3.3 Additional Assumption Details

a. The SDP Toolkit will process files of Level 0 data. Granularization of Level 0 data to any form other than as is received from SDPF or EDOS (if this does not correspond to the form required by EOS investigators) is assumed to have been performed prior to the staging of Level 0 data to the PGE. The SDP Toolkit assumes that the staged Level 0 data corresponds to the EOS investigator requested span of data, where that span of data may be based on a time range or by orbit number(s). Data grouping based on time or orbit is assumed to be done outside Toolkit calls. Where any additional granularization is performed is TBD.

b. Any further subsetting or combination of Level 0 header and quality information that is necessary as a result of granularizing Level 0 data files will have been performed prior to the staging of the data to the PGE or else is the responsibility of the science software.

c. Level 0 data files, with associated file attribute metadata, will come through the Science Data Processing Segment (SDPS) ingest data server and will be pre-staged to a given PGE.

d. The SDP Toolkit will not calculate orbit numbers for Level 0 data. Information on the orbit number corresponding to Level 0 data will be available to the SDP Toolkit through associated metadata.

e. The attribute metadata will contain all information, such as data time ranges, that are necessary for the science software to choose the correct Level 0 data file to process.

f. The Level 0 routine PGS_IO_L0_GetFileAttr will return the attribute metadata to the science software as a string buffer.

g. The science software will use the SDP Toolkit Level 0 Open functions to open Level 0 data files.

h. The Level 0 Open tools will return the Level 0 header and quality header information in the form of a string buffer. The science software will know the format and content of this header and quality information in order to unpack it.

i. The only Level 0 metadata that the SDP Toolkit Level 0 routines provide access to are found in:

    1. Attributes returned from a call to the PGS_IO_L0_GetFileAttr tool.

    2. Header and quality information physically located within a Level 0 data file which are returned from the Level 0 Open tools.

j. Packet times are returned to the science software as bit-packed data. Although the SDP Toolkit also returns the results of a time stamp conversion to a format used internally by the SDP Toolkit, additional time stamp conversions are the responsibility of the science software. Additional time stamp conversions may be made by a call to a SDP Toolkit time conversion routine such as PGS_TD_SCtime_to_UTC, which will accept the packet time stamp buffer as input and convert it into a more usable form, taking into account the platform-dependent format of the time code.

k. For each SDPF-generated Data Set File staged to a PGE, the corresponding SFDU header file will also be staged.

This page intentionally left blank.

8. Level 0 Data Simulations

8.1 Introduction

Here we present the potential application and common use of tools for generating Level 0 test data by EOS investigators, who have an obvious need for test data in the science software development. This information is given to the ECS community as a convenience; in particular, there are no SDP Toolkit requirements to generate tools for either testing Level 0 data or for the generation of simulated data. Much more discussion is needed between the EOS Project and EOS investigators regarding the development and distribution of simulated science data.

8.2 ECS Requirements for Simulated Data

ECS has a need for data containing simulated CCSDS-format packets to be used in the development and testing of the Level 0 access tools which are part of the SDP Toolkit. As described in section 7.2.6, the function PGS_IO_L0_Write_Pkt is being developed to create a simulated telemetry packet which is of CCSDS format, but which does not contain simulated science data. PGS_IO_L0_Write_Pkt is further described in Appendix A. The variable length application data field within each packet generated will contain fill or zeroed data values.

ECS has evaluated the NASA heritage code for the Generic Telemetry Simulator (GTSIM) package for use in constructing this ECS packet simulator tool. GTSIM and its current use at NASA/GSFC is described in section 8.5.3. NASA/Code 563.2 supplied ECS with GTSIM source code and sample GTSIM description files from XTE, SOHO and SWAS mission data for use as examples and to aid in our investigation. ECS has decided to not to base the design of PGS_IO_L0_Write_Pkt on the GTSIM package. It is believed that ECS has enough details of the expected packet format of TRMM, AM and PM platform telemetry to construct a limited packetizing tool. Portions of GTSIM code may be used during development of PGS_IO_L0_Write_Pkt, but the routine will not use major portions the GTSIM software package.

8.3 EOS Project Coordination of Simulated Data Generation

At the November 1993 meeting of the Science Working group for the AM Platform (SWAMP), Skip Reber of the EOSDIS Project took on the role for the coordination of test data generation for the EOS-AM platform. Discussions between the Earth Science Data Information (ESDIS) project, EOS investigators and the ECS project are ongoing to determine the best way to ensure that the EOS investigators and the ECS project are not duplicating effort.

At the Fifth Data Processing Focus Team (DPFT) meeting on April 7, 1994, Dan Marinelli of the ESDIS project gave a presentation[33] in which he proposed a Level 0 Simulation Plan. As presented, the Beta delivery of this proposed plan would include the delivery of the ECS-developed packet simulator functionality described in section 7.2.4 to EOS investigators. As performed by the ESDIS project and EOS investigators, subsequent stages of this Level 0 simulation plan involves the increasing use and complexity of simulated instrument science data. This would involve testing in the Science Computing Facility (SCF) environment that includes simulated telemetry and communications errors.

8.4 Existing Institutional Test Efforts

8.4.1 TRMM Test and Training Simulator (TTS)

The TRMM Test and Training Simulator (TTS) is a NASA Code 515 project. The TTS is designed to provide training for the TRMM Flight Operations Team, but will also be part of pre launch activities such as the Mission Operations Center (MOC) telemetry and command data base validation and checkout of the TRMM ground system software capabilities and network interfaces.

The Tropical Rainfall Measuring Mission (TRMM) Test and Training Simulator (TTS) Functional Requirements Document (FRD)[34] states that the TTS will simulate the onboard spacecraft clock functions, including the simulation of the onboard mission elapsed time (MET) as a 64-bit counter with a one microsecond resolution. The TTS FRD also states that the TTS is not required to provide dynamic instrument and science simulation. However, the TTS FRD does state that the TTS is required to interleave realistic CERES, LIS, PR, VIRS and TMI instrument data provided on tape by the science and/or TRMM integration and test team. The schedule for operational use of the TTS is TBD.

8.4.2 EOS Test System (ETS)

The EOS Test System (ETS) is a NASA Code 515 project that is still in the requirements gathering stage. The EOSDIS Test System (ETS) Operations Concept document[35] states that this project will "provide testing capabilities for the ground system supporting EOS spacecraft." ETS will simulate the output of EDOS Production Data Sets and the transfer of them to ECS. The first stage of ETS is not expected to be up and running before June 1996[36].

8.4.3 Generic Telemetry Simulator (GTSIM)

SDPF (Pacor/DDF) is currently using the Generic Telemetry Simulator (GTSIM) software package for its own system testing. Pacor will support the TRMM mission by developing TRMM-specific input specifications files for use with the GTSIM package. These TRMM-specific input specifications files do not yet exist but will be constructed by GTSIM developers based on data format details supplied by the TRMM science teams. The time frame for this support of the TRMM mission is not expected to be earlier than January 1996. The GTSIM software can then be used by TRMM science teams to simulate the TRMM telemetry data to be provided by SDPF. The simulated data will have the structure of CCSDS format telemetry packets, but no science content will be simulated. The Pacor Project Manager is expected to be the point of contact for the TRMM science teams in any use of the GTSIM software[37].

8.4.4 SDP Toolkit Packet Formatter

ECS is working closely with GTSIM developers and with Pacor in order to avoid duplication of effort. Although Pacor and GTSIM will provide support for the TRMM mission, ECS intends to provide packet simulation capabilities that will be available to EOS investigators in a much earlier time frame. As a first cut, the Toolkit will contain software which will create data files containing packets in the format described in Section 4.2 of this document. Packet time stamps will be generated and made to correspond to simulated platform orbit and attitude data generated by the Toolkit (Section 5.4). Instrument teams can insert simulated instrument data into the packets for testing Level 1 data production code.

This page intentionally left blank.

9. Open Issues

The purpose of this section is to provide an overview of some of the many open issues and concerns related to Level 0 data formats and ECS processing issues that this White Paper attempts to address. The previous sections of this White Paper provide more in-depth discussions of these and other issues.

- More information is needed regarding what granule size requirements have been relayed from EOS investigators to the EDOS.
- ECS intends that the use of the SDP Toolkit routines described here will fix the interface of the science software to Level 0 instrument data, but that the underlying software will not be fixed and will change as processing needs and requirements change. Proposed design changes are presented in this White Paper and are due to:
    a. the SDPF- and EDOS-generated Level 0 quality metadata being inherently tied to a physical data file, not a data granule,
    b. concerns over memory allocation,
    c. and a lack of information on the form and content of metadata generated internally by ECS.
- ECS needs further clarification from EDOS as to what their data packaging plans and/or limitations are. Also, it is crucial that content and format information be made available for EDOS-generated Level 0 Production Data Sets and for the header and quality metadata parameters that will be located within them.
- More detailed information is needed on how EOS investigators anticipate accessing the metadata physically located within Level 0 data files and also how they envision the science software using this information in science production processing. A preliminary definition of core metadata content for the ECS Project is due by December, 1994. This definition may change as the system matures. Toolkit access methods to this metadata, should not change, however.
- A singular definition of the term "granule" is needed. At the very least, EOS investigators, EDOS, ESDIS and ECS should all be aware the differences in how each group defines a data granule and what the processing implications are. The SDP Toolkit assumption is that physical files are the staged input to the PGE. The time extent of these staged files may not be identical to (but will bound) the time extent of the Level 1A processing request.
- ECS needs to confirm with the Flight Dynamics Facility the availability and ingest interfaces for the various orbit and attitude products discussed here as inputs to Level 0 data processing.
- Resolution is needed between the ECS and TRMM Projects over the concerns about the format of the packet time stamps appearing in TRMM-platform Level 0 data.

- Better coordination is needed among the ESDIS Project, ECS and EOS investigators to avoid duplication of effort in dealing with the issues presented here.

It is hoped that the issues raised in this White Paper will provide a basis for increased discussion between ECS and the EOS investigators on the necessary environment for processing Level 0 data. ECS will be working more closely with the EOS science software developers to better understand their intended usage of the SDP Toolkit in Level 0 processing.

Appendix A. Level 0 Access Function Descriptions

A.1 Introduction

These Level 0 access tools will be used to open and read data from Level 0 data files. These tools allow access to Level 0 data files whether they have been generated and formatted by EDOS for AM and PM platform data, or by the Science Data Processing Facility (SDPF) (Pacor/DDF) for TRMM platform data. (Pacor is an abbreviation for packet processor; DDF is Data Distribution Facility.)

A brief summary of the functions of these tools follows.

The Level 0 file initialization tool (PGS_IO_L0_Init) set up global variables which allow the SDP Toolkit to provide the science software with time-ordered access to file attributes, even if the Level 0 data files were not staged to a PGE in time order.

Level 0 file initialization (PGS_IO_L0_Init) and file attribute tools (PGS_IO_L0_GetFileAttr) both return useful information retrieved from ECS-internal metadata regarding the staged Level 0 files to the science software.

Level 0 File Open tools (PGS_IO_L0_SDPF_Open & PGS_IO_L0_EDOS_Open) will return the following to the science software:

a. buffers containing the SDPF- or EDOS-specific header and quality information that are physically located in the Level 0 data set file. The header and accounting information will include parameters such as the number of data packets and time range of the data. The Q/A information will include such items as listings of missing packets or Reed-Solomon errors.

b. the file handle (i.e., physical file name) of the Level 0 data files that the user has input a logical file name and version number for.

The Level 0 read tool PGS_IO_L0_Read_Pkt reads a single telemetry packet and returns:

a. the parameters contained in the packet's primary and secondary header. These values are returned to the science software. Some of these values, such as the time stamp and user flags, must be unpacked and interpreted by the science software, possibly using other SDP Toolkit routines. However, this routine will also return the packet time stamp converted to an internal format. The SDP Toolkit needs this time stamp converted to an internal format (TAI, as seconds since 12AM UTC 01-01-1993) so that it can more easily track the data packet times internally. This converted packet time value is being returned to the science software to avoid potential duplication of effort and processing.

b. a buffer containing the content of variable-length packet application data field.

Additionally, a tool (PGS_IO_L0_Write_Pkt) is being proposed which will generate CCSDS-formatted Level 0 packets, given certain TBD user input parameters. Although there is currently no SDP requirement for a Level 0 write tool, it is certain that such a tool will be necessary for effective generation of test data at the SCFs.

A.2 PGS_IO_L0_Init

Initialize Level 0 Processing

NAME: **PGS_IO_L0_Init( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_Init(
        PGSt_PC_Logical     file_logical,
        PGSt_integer        *num_files,
        PGSt_double         *time1_req,
        PGSt_double         *time2_req,
        PGSt_double         *time_first,
        PGSt_double         *time_last,
        PGSt_double         *time_start,
        PGSt_double         *time_end,
        TBD additional parameters   );
```

FORTRAN:

integer function pgs_io_l0_init (file_logical, num_files, time1_req, time2_req, time_first, time_last, time_start, time_end, *TBD*)

```
        integer             file_logical
        integer             num_files
        double precision    time1_req
        double precision    time2_req
        double precision    time_first
        double precision    time_last
        double precision    time_start
        double precision    time_end
        (TBD additional parameters)
```

INPUTS: file_logical - The user defined file logical for the Level 0 data file (EDOS PDS file or SDPF Data Set File).

OUTPUTS: num_files - Number of physical Level 0 data files staged to PGE. This is determined by an internal SDP Toolkit call to the function PGS_PC_GetNumberOfFiles.

time1_req - Start time of the Level 0 data span as previously requested by EOS investigators (through the ECS planner/scheduler) to be staged to the PGE. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

time2_req - End time of the Level 0 data span as previously requested by EOS investigators (through the ECS planner/scheduler) to be staged to the PGE. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

time_first - Start time of the staged Level 0 data. This is determined by an internal SDP Toolkit call to the function PGS_PC_GetFileAttr. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

time_last - End time of the staged Level 0 data. This is determined by an internal SDP Toolkit call to the function PGS_PC_GetFileAttr. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

time_start - Start time of the user-defined time range of staged Level 0 data to be processed. This value can subsequently be reset by the science software, but is returned here to the user with a default value equal to the value *time_first.* This value is used by the SDP Toolkit as it internally tracks the times of individual packets in relation to the user defined data range. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

time_end - End time of the user-defined time range of staged Level 0 data to be processed. This value can subsequently be reset by the science software, but is returned here to the user with a default value equal to the value *time_last.* This value is used by the SDP Toolkit as it internally tracks the times of individual packets in relation to the user defined data range. This value will be in TAI, as real continuous seconds since 12AM UTC 01-01-1993.

TBD additional output parameters

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX
PGSIO_E_L0_FILE_NOEXIST
PGSIO_E_L0_REFERENCE_FAILURE

NOTES: This routine initializes various global variables necessary for the proper functioning of the Level 0 Open and Read packets tool. This routine must be called prior to any accessing of Level 0 metadata or data files. It is intended that returning select metadata to the science software using PGS_IO_L0_Init will reduce the need for individual calls by the science software to the routine PGS_IO_L0_GetFileAttr for each physical Level 0 data file.

EXAMPLES: Below is an example (for an SDPF-generated Level 0 Data Set File) of how the function PGS_IO_L0_Init can be used to initialize Level 0 processing.

C:

```
PGSt_PC_Logical   prodID_2;   /* file_logical for SDPF Data Set File */
PGSt_integer      version;    /* internal counter for multiple physical
                                 files*/
PGSt_IO_Gen_FileHandle  *file_handle;     /* file handle of the SDPF Data
                                 Set File */
PGSt_SMF_Status   returnStatus;     /* SDP Toolkit status message */
PGSt_integer      num_files;  /* variable for # of staged SDPF Data Set
                                 Files */
PGSt_double       time1_req;  /* variable for start time of time range
                                 requested through ECS planner/scheduler */
PGSt_double       time2_req;  /* variable for end time of time range
                                 requested through ECS planner/scheduler */
PGSt_double       time_first; /* variable for start time of actual time range
                                 of staged data */
PGSt_double       time_last;  /* variable for end time of actual time range
                                 of staged data */

PGSt_double       time_start; /* user defined variable for start time of
                                 range of data to be processed */
PGSt_double       time_end;   /* user defined variable for end time of
                                 range of data to be processed */

/* initialize the user-defined logical identifier for SDPF Data Set File */

prodID_2 = 102;

/* Call Level 0 processing initialization routine which loads various global
   variables; PGS_IO_L0_Init also returns the number of data files and the
   time range of the staged data */

returnStatus = PGS_IO_L0_Init (prodID_2, &num_files, &time1_req, &time2_req,
                  &time_first, &time_last, &time_start, &time_end, TBD );

if (returnStatus != PGS_S_SUCCESS)
{
      return (returnStatus);
}
else
{
      /* If necessary, re-set the start and end times for the Level 0 packet
         reads to follow; these values can be set by the user based
         on values returned from the PGS_IO_L0_Init routine for the actual
         start and end times of the staged data (i.e.time_first and time_last,
         which are the default values of time_start and time_end)  */

      time_start = ****;
      time_end = ****;
}
```

Fortran:

```
      integer           prodid_1
      integer           prodid_2
      integer           version
      integer           returnstatus
      integer           num_files
      character*(TBD)   fileattr
      integer           pgs_io_l0_init
      double precision  time1_req
      double precision  time2_req
      double precision  time_first
      double precision  time_last
      double precision  time_start
      double precision  time_end

C     Initialize the user-defined logical identifiers for the SFDU header file
C     and the SDPF Data Set File

      prodid_1 = 101
      prodid_2 = 102

C     Call Level 0 processing initialization routine which loads various
C     global variables; pgs_io_l0_init also returns the number of data files
C     and the time range of the staged data

      returnstatus = pgs_io_l0_init(prodid_2, num_files, time1_req,
     *                  time2_req, time_first, time_last, time_start,
     *                  time_end, TBD )

      if (returnstatus .ne. pgs_s_success) then

            go to 5000

      else

C           If necessary, re-set the start and end times for the Level 0
C           packet reads to follow; these values can be set by the user
C           based on values returned from the pgs_io_l0_init routine for the
C           actual start and end times of the staged data (i.e.time_first
C           and time_last, which are the default values of time_start and
C           time_end)

            time_start = ****
            time_end = ****
            .
            .
            .

      end if

            .
            .
            .

 5000 <error handling goes here>
```

DETAILS: Transparent to the science software, this routine will make a call to the PGS_PC_GetNumberOfFiles to determine the number of staged physical files for the input logical file identifier. For each of these physical files, this routine will also make internal calls to PGS_PC_GetFileAttr in order to map the time-ordered sequence of the physical files and load the information into global variables, since they may not be staged in time order. In addition to retrieving selected metadata parameters from the PC table file attributes for internal SDP Toolkit usage, the internal calls to PGS_PC_GetFileAttr will also retrieve and return to the science software a TBD number of file attributes.

A.3 PGS_IO_L0_GetFileAttr

Return Time-Ordered Level 0 File Attributes

NAME: **PGS_IO_L0_GetFileAttr( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_GetFileAttr(
        PGSt_PC_Logical     file_logical,
        PGSt_integer        file_version,
        char                *fileAttr );
```

FORTRAN:

```
integer function pgs_io_l0_getfileattr (file_logical, file_version, fileattr)

        integer             file_logical
        integer             file_version
        character*(TBD)     fileattr
```

INPUTS: file_logical - A constant passed to this routine for mapping to a physical file. The input value *file_version* is also required in order to uniquely identify a staged data file before file attributes can be returned.

file_version - A constant passed to this routine for mapping of the logical file name to a physical file name. This parameter is necessary due to the possibility of multiple physical files of Level 0 data being staged to a given PGE. This is a value that is determined by the results of a previous call to the PGS_IO_L0_Init function.

OUTPUTS: fileAttr - String of file attribute metadata retrieved from a PC table file attribute file.

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX
PGSIO_E_L0_FILE_NOEXIST
PGSIO_E_L0_REFERENCE_FAILURE

NOTES: This function provides transparent access to time-ordered file attributes for Level 0 data files. The file attribute information is returned to the science software as a string; the individual parameters are not unpacked by the SDP

Toolkit. This function is a wrapper on the SDP Toolkit function PGS_PC_GetFileAttr. This wrapper is necessary since the version numbers of the Level 0 data files staged to a PGE may not be in time order. This function internally translates the input file version number to the time-ordered file version number and then retrieves the file attributes for that physical file. It is intended that previously returning select metadata to the science software using PGS_IO_L0_Init will reduce the need for individual calls by the science software to PGS_IO_L0_GetFileAttr for each physical Level 0 data file.

EXAMPLES: Below is an example (for an SDPF-generated Level 0 Data Set File) of how the function PGS_IO_L0_GetFileAttr can be retrieve file attributes for a given Level 0 file that has been staged to PGE.

C:

```
PGSt_PC_Logical   prodID_1;   /* file_logical for detached SFDU header file*/
PGSt_PC_Logical   prodID_2;   /* file_logical for SDPF Data Set File */
PGSt_integer      version;    /* internal counter for multiple physical
                                 files*/
PGSt_SMF_Status   returnStatus;     /* SDP Toolkit status message */
PGSt_integer      num_files;  /* variable for # of staged SDPF Data Set
                                 Files */
PGSt_double       time1_req;  /* variable for start time of time range
                                 requested through ECS planner/scheduler */
PGSt_double       time2_req;  /* variable for end time of time range
                                 requested through ECS planner/scheduler */
PGSt_double       time_first; /* variable for start time of actual time range
                                 of staged data*/
PGSt_double       time_last;  /* variable for end time of actual time range of
                                 staged data*/
char              fileAttr[PGSd_IO_MAX_FILEATTR];     /* buffer of file
                                 attributes */

/* initialize the user-defined logical identifiers for the SFDU header file
   and the SDPF Data Set File */

prodID_1 = 101;
prodID_2 = 102;

/* call Level 0 processing initialization routine which loads various global
   variables; PGS_IO_L0_Init also returns the number of data files and the
   time range of the staged data */

returnStatus = PGS_IO_L0_Init (prodID_2, &num_files, &time1_req, &time2_req,
                  &time_first, &time_last, &time_start, &time_end, TBD );
      .
      .
      .

/* loop through and process in turn each physical Level 0 data file */

for (version = 1; version <= num_files; version++)
{
      /* if necessary, read in file attributes for the staged physical file
         corresponding to this version's SFDU header file; because the version
         numbers in the PC table may not be time ordered, the Level 0 wrapper
         of PGS_PC_GetFileAttr routine gives the science software
         transparent access to time ordered file attributes */

      returnStatus = PGS_IO_L0_GetFileAttr(prodID_1, version, fileAttr);

      if (returnStatus != PGS_S_SUCCESS)
      {
            return (returnStatus);
      }
      else
      {

            /* unpack fileAttr string buffer for this SFDU header file */
            /* interpret and store TBD file attributes such as time range of
               file as necessary*/

      }

      /* if necessary, read in file attributes for the staged physical file
         corresponding to this version's SDPF Data Set File; because the
         version numbers in the PC table may not be time ordered, the Level 0
         wrapper of PGS_PC_GetFileAttr routine gives the science software
         transparent access to time ordered file attributes */

      returnStatus = PGS_IO_L0_GetFileAttr(prodID_2, version, fileAttr);

      if (returnStatus != PGS_S_SUCCESS)
      {
            return (returnStatus);
      }
      else
      {

            /* unpack fileAttr string buffer for this SDPF Data Set File */
            /* interpret and store TBD file attributes such as time range of
               file as necessary*/

      }

}
```

Fortran:

```
      integer             prodid_1
      integer             prodid_2
      integer             version
      integer             returnstatus
      integer             num_files
      character*(TBD)     fileattr
      integer             pgs_io_l0_init
      integer             pgs_io_l0_getfileattr
      double precision    time1_req
      double precision    time2_req
      double precision    time_first
      double precision    time_last
      double precision    time_start
      double precision    time_end
```

```
C     Initialize the user-defined logical identifiers for the SFDU header file
C     and the SDPF Data Set File

      prodid_1 = 101
      prodid_2 = 102

C     Call Level 0 processing initialization routine which loads various
C     global variables; pgs_io_l0_init also returns the number of data files
C     and the time range of the staged data

      returnstatus = pgs_io_l0_init(prodid_2, num_files, time1_req,
     *                  time2_req, time_first, time_last, time_start,
     *                  time_end, TBD )

C     Loop through and process in turn each physical Level 0 data file

      do 100 version = 1, num_files

C           If necessary, read in file attributes for the staged physical file
C           corresponding to this version's SFDU header file; because the
C           version numbers in the PC table may not be time ordered, the
C           Level 0 wrapper of pgs_pc_getfileattr routine gives the science
C           software transparent access to time ordered file attributes

            returnstatus = pgs_io_l0_getfileattr(prodid_1, version, fileattr)

            if (returnstatus .ne. pgs_s_success) then

                  go to 5000

            else

                  <unpack fileattr string buffer for this SFDU header file>
                  <interpret and store TBD file attributes such as time range
                    of file as necessary>

            end if

C           If necessary, read in file attributes for the staged physical file
C           corresponding to this version's SDPF Data Set File; because the
C           version numbers in the PC table may not be time ordered, the
C           Level 0 wrapper of pgs_pc_getfileattr routine gives the science
C           software transparent access to time ordered file attributes

            returnstatus = pgs_io_l0_getfileattr(prodid_2, version, fileattr)

            if (returnstatus .ne. pgs_s_success) then

                  go to 5000

            else

                  <unpack fileattr string buffer for this SDPF Data Set File>
                  <interpret and store TBD file attributes such as time range
                    of file as necessary>

            end if

            .
            .
            .
```

```
 100  Continue
            .
            .
            .
 5000 <error handling goes here>
```

DETAILS: The function PGS_IO_L0_Init must have been called prior to PGS_IO_L0_GetFileAttr in order to set up the global variables that allow the version number input here to be translated by the SDP Toolkit into the time-ordered file version number.

A.4 PGS_IO_L0_SDPF_Open

Open SDPF-Generated Level 0 File

NAME: **PGS_IO_L0_SDPF_Open( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_SDPF_Open (
        PGSt_PC_Logical         file_logical_1,
        PGSt_PC_Logical         file_logical_2,
        PGSt_integer            file_version,
        char                    *sdpf_hdr_1,
        char                    *sdpf_hdr_2,
        char                    *sdpf_qual,
        PGSt_IO_Gen_FileHandle  **file_handle);
```

FORTRAN:

```
integer function pgs_io_l0_sdpf_open(file_logical_1, file_logical_2,
file_version, sdpf_hdr_1, sdpf_hdr_2, sdpf_qual, file_handle)

        integer             file_logical_1
        integer             file_logical_2
        integer             file_version
        character*(TBD)     sdpf_hdr_1
        character*(TBD)     sdpf_hdr_2
        character*(TBD)     sdpf_qual
        integer             file_handle
```

DESCRIPTION: This tool may be used to open a Level 0 data set generated by SDPF (Pacor/DDF) for TRMM platform instrument telemetry and return the file handle of the SDPF Data Set File. This tool returns a buffer containing SDPF-formatted Level 0 header (i.e., accounting) information and quality metadata information extracted from the physical Level 0 data file. This information on the time extent, number and quality of the Level 0 data packets will be used by the science software when preparing to use a SDP Toolkit Level 0 read routine.

INPUTS:

file_logical_1 - A constant identifying the logical file corresponding to the SFDU detached header file. This value is passed to this routine for mapping to a physical file. The input value *file_version* is also required in order to uniquely identify this staged data file before a file handle can be returned.

file_logical_2 - A constant identifying the logical file corresponding to the Data Set file. This value is passed to this routine for mapping to a physical file.

file_version - A constant identifying the version number corresponding to this logical file set (i.e. detached SFDU header file and SDPF Data Set File). This is a value that is determined from the results of a previous call to the PGS_IO_L0_Init function. This value is passed to this routine for mapping of the logical file name to a physical file name. This is necessary due to the possibility of multiple physical files being staged to a given PGE.

OUTPUTS:

sdpf_hdr_1 - The C and FORTRAN implementations of this routine will output the buffer *sdpf_hdr_1* containing a TBD list of parameters extracted from the SDPF detached SFDU header file.

sdpf_hdr_2 - The C and FORTRAN implementations of this routine will output the buffer *sdpf_hdr_2* containing a TBD list of parameters extracted from the SDPF Data Set File header.

The following is a list of those parameters expected to be included in the SDPF Level 0 header information. It is not meant to represent a complete or definitive list, but is based on the best available information.

- Number of packets in file - From a 32-bit field which contains the actual number of source data units in the Level 0 data set.
- Spacecraft clock, first source data unit (i.e. packet) - From a 72-bit field containing the Universal Time Coordinate (UTC) equivalent of the spacecraft clock reading, rounding to the nearest microsecond and encoded to parallel binary format (PB-5) time code, at the beginning of the first source data unit of the Level 0 data set.
- Spacecraft clock, last source data unit - From a 72-bit field containing the UTC equivalent of the spacecraft clock reading, rounding to the nearest microsecond and encoded to parallel binary format (PB-5) time code, at the beginning of the last source data unit of the Level 0 data set.
- Processing Option Flag - From an 8-bit field, each bit of which is used to indicate whether an optional process was performed. The individual bits are defined as follows:

Bits 1,2	Reserved
Bit 3	Redundant Data Deleted
Bit 4	Telemetry Frame Cyclic Redundancy Code (CRC) decoding
Bit 5	Virtual Channel Data Unit (VCDU) Header Control decoding
Bit 6	Data Merging Performed
Bit 7	Reed-Solomon decoding
Bit 8	Data Reversal Performed

- Data Type Flag - From an 8-bit field, each bit of which indicates the general type of data contained in the output stream. Routine Production Data is indicated when Bit 1 is set. Quick-Look data is indicated when Bit 2 is set.

- Time of Receipt at originating node - From a 56-bit field containing the time to the nearest millisecond, in PB-5 format, at which the last telemetry acquisition session containing the last downlinked source data unit was terminated at Pacor.

- Number of Offsets to Quality and Accounting Capsule (QAC) - From a 32-bit field indicating the position of the QAC counted from the last bit of the Data Set File Header.

sdpf_qual - The C and FORTRAN implementations of this routine will output the buffer *sdpf_qual* containing a TBD list of quality information parameters extracted from the SDPF Data Set File. Existing SDPF documentation states that Level 0 data sets will contain a Quality Accounting Capsule (QAC) and a Missing Data Units Entry (MDUE). The following is a list of those parameters expected to be included in the SDPF Level 0 QAC and MDUE information based on currently available information.

The QAC will contain the following information:

- Length of QAC - From a 32-bit field specifying the length, in bytes, of the QAC list for the Level 0 data set file.

- Offset of Data Unit (i.e. packet) in Data Set - From a 32-bit field indicating the position of the erred data unit in the data set counted from the last bit of the Data Set File Header.

- Data Unit Sequence Count - From a 16-bit field containing a count indicating the (packet) sequence number of the Source Data Unit.

- Error Type Flags - From an 8-bit field indicating the types of error(s) associated with the packet, as defined below:

Bit 0	Not Used
Bit 1	Reed-Solomon Header Errors
Bit 2	Data Unit Length Code Wrong
Bit 3	Reed-Solomon Frame Errors
Bit 4	CRC Frame Errors
Bit 5	Data Unit Sequence Count Error/Discontinuity
Bit 6	Detected Frame Errors during generation of packet
Bit 7	Data Unit Contains Fill Data

- Count of Segments with CRC/RS Errors - From an 8-bit field specifying the number of segments of the data unit that were from frames with errors detected by frame CRC check or Reed-Solomon error - correction process.
- Data Unit Fill Start Location - From a 16-bit field indicating the byte location (1-n) of the start of fill in the source data unit. A value of zero indicates that there is no fill.
- The MDUE follows the QAC and will contain a missing packet list for the Level 0 data set, identified by packet sequence numbers.

file_handle - The file handle returned to the science software for the SDPF Data Set file containing the Level 0 data packets.

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX
PGSIO_E_L0_FILE_NOEXIST
PGSIO_E_L0_REFERENCE_FAILURE

NOTES: While this tool will return a buffer containing the header and quality information parameters stored in Level 0 data sets, it is the responsibility of the science software to unpack and interpret the parameters within the buffer. It is expected that further details on the structure and content of SDPF-generated Level 0 files will be available by December 1994.

EXAMPLES: Below is an example of how the function PGS_IO_L0_SDPF_Open can be used to open an SDPF-generated Level 0 Data Set File.

C:

```
PGSt_PC_Logical   prodID_1;   /* file_logical for detached SFDU header file*/
PGSt_PC_Logical   prodID_2;   /* file_logical for SDPF Data Set File */
PGSt_integer      version;    /* internal counter for multiple physical
                                 files*/
PGSt_IO_Gen_FileHandle  *file_handle;     /* pointer to file handle */
PGSt_SMF_Status   returnStatus;     /* SDP Toolkit status message */
char              sdpf_hdr_1[PGSd_IO_MAX_SDPFHDR1];   /* buffer
                                 containing SFDU header data */
char              sdpf_hdr_2[PGSd_IO_MAX_SDPFHDR2];   /* buffer
```

```
                              containing SDPF Data Set File header data */
char              sdpf_qual[PGSd_IO_MAX_SDPFQUAL];    /* buffer
                              containing SDPF Data Set File quality data */
PGSt_integer      num_files;  /* variable for # of staged SDPF Data Set
                              Files */

/* initialize the user-defined logical identifiers for the SFDU header file
   and the SDPF Data Set File */

prodID_1 = 101;
prodID_2 = 102;

/* loop through and process in turn each physical file of the SDPF-generated
   Level 0 data file sets */

for (version = 1; version <= num_files; version++)
{
      /* for this version's physical file set (i.e. SFDU header and Data Set
         File), call a routine to open the SDPF-generated files, return the
         header and quality information in those files and return the file
         handle of the SDPF Data Set File */

      returnStatus = PGS_IO_L0_SDPF_Open (prodID_1, prodID_2, version,
                        sdpf_hdr_1, sdpf_hdr_2, sdpf_qual, &file_handle);

      if (returnStatus != PGS_S_SUCCESS)
      {
            return (returnStatus);
      }
      else
      {

            /* unpack header parameters within sdpf_hdr_1 string */
            /* unpack header parameters within sdpf_hdr_2 string */
            /* unpack quality parameters within sdpf_qual string */
            /* interpret header and quality data and set internal variables to
               track if the science software must flag and/or avoid any data
               sets or packet data as "bad" */

      }
}
```

FORTRAN:

```
      integer           prodid_1
      integer           prodid_2
      integer           version
      integer           returnstatus
      integer           num_files
      character*(TBD)   fileattr
      integer           pgs_io_l0_sdpf_open

C     Initialize the user-defined logical identifiers for the SFDU header file
C     and the SDPF Data Set File

      prodid_1 = 101
      prodid_2 = 102

C     Loop through and process in turn each physical Level 0 data file of the
C     SDPF-generated Level 0 data file sets
```

```
      do 200 version = 1, num_files

C           For this version's physical file set (i.e. SFDU header and Data
C           Set File), call a routine to open the SDPF-generated files, return
C           the header and quality information in those files and return the
C           file handle of the SDPF Data Set File

            returnstatus = pgs_io_l0_sdpf_open (prodid_1, prodid_2, version,
     *                        sdpf_hdr_1, sdpf_hdr_2, sdpf_qual, file_handle)

            if (returnstatus .ne. pgs_s_success) then

                  go to 5000

            else

            < unpack header parameters within sdpf_hdr_1 string >
            < unpack header parameters within sdpf_hdr_2 string >
            < unpack quality parameters within sdpf_qual string >
            < interpret header and quality data and set internal variables to
               track if the science software must flag and/or avoid any data
               sets or packet data as "bad" >

            end if

            .
            .
            .

 200  Continue
            .
            .
            .

 5000 <error handling goes here>
```

DETAILS: If multiple physical file sets (i.e. SFDU detached header and data set file) of Level 0 data are staged to a PGE, this function must be called for each physical Level 0 file set from which header and quality information is desired. This Level 0 Open tool must be called before the tool PGS_IO_L0_Read_Pkt can be used to read packet data for a given SDPF-generated Level 0 data file set. This tool will call the function PGS_IO_Gen_Open in order to map input logical file names and file version numbers to physical file names and to open the files for read-only access. The function PGS_IO_L0_Init must have been called prior to PGS_IO_L0_SDPF_Open in order to set up the global variables that allow the version number input here to be translated by the SDP Toolkit into the time-ordered file version number.

A.5 PGS_IO_L0_EDOS_Open

Open EDOS-Generated Level 0 File

NAME: **PGS_IO_L0_EDOS_Open( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_EDOS_Open (
          PGSt_PC_Logical           file_logical,
          PGSt_integer              file_version,
          char                      *edos_hdr_qual,
          PGSt_IO_Gen_FileHandle    **file_handle)
```

FORTRAN:

```
integer function pgs_io_l0_edos_open(file_logical, file_version,
edos_hdr_qual, file_handle)

          integer              file_logical
          integer              file_version
          character*(TBD)      edos_hdr_qual
          integer              file_handle
```

DESCRIPTION: This tool may be used to open a Level 0 data set generated by EDOS for AM and PM platform data. This tool returns a buffer containing EDOS formatted Level 0 header (i.e., accounting) information and quality metadata information extracted from the physical Level 0 data file. This information on the time extent, number and quality of the Level 0 data packets will be used by the science software when preparing to use the SDP Toolkit Level 0 read routine PGS_IO_L0_Read_Pkt.

INPUTS: file_logical - A constant passed to this routine for mapping to a physical file. The input value *file_version* is also required in order to uniquely identify a staged data file before a file handle can be returned.

file_version - A constant passed to this routine for mapping of the logical file name to a physical file name. This parameter is necessary due to the possibility of multiple physical files of Level 0 data being staged to a given PGE. This is a value that is determined by the results of a previous call to the PGS_IO_L0_Init function.

OUTPUTS: edos_hdr_qual - The C and FORTRAN implementations of this routine will output the buffer *edos_hdr_qual* containing a TBD list of parameters extracted from header and quality information in the EDOS Level 0 file. This tool will not unpack the individual parameters within this buffer. Current EDOS documentation indicates that the information stored in the Level 0 data sets will, at a minimum, include the following parameters (this is not meant to represent a complete or definitive list, but is based on the available information).

- Production Data Set ID - A unique identifier for each Level 0 data set generated by EDOS.
- Applications Process ID (APID) associated with the Level 0 data set.
- Virtual Channel Data Unit (VCDU)-ID associated with the Applications Process Identifier.
- List of scheduled Tracking and Date Relay Satellite System (TDRSS) Service Session (TSS) start and stop times for the Level 0 data set.
- Time of Level 0 data set completion.
- EDOS Service Header time and date annotation of the first and last packets in the Level 0 data set.
- Level 0 data set size (in octets).
- Consultative Committee on Space Data Systems (CCSDS) binary time codes stored in the secondary header of the first and the last packets in the Level 0 data set.
- Packet sequence counts of the first and last packets in the Level 0 data set.
- Count of packets from VCDUs with errors corrected by Reed-Solomon decoding.
- List of missing packet sequence counts for the Level 0 data set.
- List of packets containing EDOS generated fill data, including location of first fill octet for each packet.
- Count of packets containing EDOS generated fill data.
- Count of octets of EDOS generated fill data.
- Count of packets with discrepancies between packet header length code and actual packet length.

- Count of packets with sequence count discontinuities.

file_handle - The file handle returned to the science software for the Level 0 data file.

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX
PGSIO_E_L0_FILE_NOEXIST
PGSIO_E_L0_REFERENCE_FAILURE

NOTES: While this tool will return a buffer containing the header and quality information parameters stored in Level 0 data sets, it is the responsibility of the science software to unpack and interpret the parameters within the buffer. Also, the specific parameters and their formats are TBD. Detailed information is not currently available regarding the specific parameter lists, formats or locations within the EDOS-generated Level 0 data sets. It is expected that details on the structure and content of EDOS-generated Level 0 files will be available by October 1994.

EXAMPLES: Below is an example of how the function PGS_IO_L0_EDOS_Open can be used to open an EDOS-generated Level 0 PDS file.

C:

```
PGSt_PC_Logical   prodID;     /* file_logical for EDOS PDS file */
PGSt_integer      version;    /* internal counter for multiple physical
                                 files */
PGSt_IO_Gen_FileHandle  *file_handle;     /* pointer to file handle */
PGSt_SMF_Status   returnStatus;     /* SDP Toolkit status message */
char              edos_hdr_qual[PGSd_IO_MAX_EDOSHDR]; /* buffer
                                 containing EDOS Header and Quality data */
PGSt_integer      num_files;  /* variable for # of staged EDOS PDS files */

/* initialize the user-defined logical identifier for the EDOS PDS file */

prodID = 201;

/* loop through and process in turn each physical Level 0 data file */

for (version = 1; version <= num_files; version++)
{

      /* for each desired EDOS PDS file, call a routine to open the EDOS-
         generated files, return the file handle of the EDOS PDS file and also
         return header and quality information located in the EDOS PDS  */

      returnStatus = PGS_IO_L0_EDOS_Open (prodID, version, edos_hdr_qual,
                     &file_handle);

      if (returnStatus != PGS_S_SUCCESS)
      {
            return (returnStatus);
      }
      else
      {

            /* unpack quality parameters within edos_hdr_qual string */
            /* interpret header and quality data and set internal variables to
               track if the science software must flag and/or avoid any data
               sets or packet data as "bad" */

      }

}
```

FORTRAN:

```
      integer           prodid
      integer           version
      integer           returnstatus
      integer           num_files
      character*(TBD)   fileattr
      integer           pgs_io_l0_edos_open
      double precision  time1_req
      double precision  time2_req
      double precision  time_first
      double precision  time_last
      double precision  time_start
      double precision  time_end

C     Initialize the user-defined logical identifier for the EDOS PDS file

      prodid = 201

C     Loop through and process in turn each physical Level 0 data file

      do 300 version = 1, num_files

C           For each desired EDOS PDS file, call a routine to open the EDOS-
C           generated files, return the file handle of the EDOS PDS file and
C           also return header and quality information located in the EDOS PDS

            returnstatus = pgs_io_l0_edos_open (prodid, version,
     *                           edos_hdr_qual, file_handle)

            if (returnstatus .ne. pgs_s_success) then

                  go to 5000

            else

            < unpack quality parameters within edos_hdr_qual string >
            < interpret header and quality data and set internal variables to
               track if the science software must flag and/or avoid any data
               sets or packet data as "bad" >

            end if

            .
            .
            .

 300  Continue
            .
```

```
              .
              .
 5000 <error handling goes here>
```

DETAILS: If multiple physical files of EDOS PDS Level 0 data are staged to a given PGE, this function must be called for each physical Level 0 file from which header and quality information is desired. This Level 0 Open tool must be called before the tool PGS_IO_L0_Read_Pkt can be used to read packet data for a given EDOS-generated Level 0 data file. This tool will call the function PGS_IO_Gen_Open in order to map input logical file names and file version numbers to physical file names and to open the files for read-only access. The function PGS_IO_L0_Init must have been called prior to PGS_IO_L0_EDOS_Open in order to set up the global variables that allow the version number input here to be translated by the SDP Toolkit into the time-ordered file version number.

A.6 PGS_IO_L0_Read_Pkt

Read Level 0 Packet

NAME: **PGS_IO_L0_Read_Pkt( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_Read_Pkt(
        PGSt_IO_Gen_FileHandle    *file_handle,
        PGSt_scTag                spacecraftTag,
        char                      *pkt_idflags,
        PGSt_integer              *pkt_apid,
        char                      *pkt_seqflg,
        PGSt_integer              *pkt_seqcount,
        PGSt_integer              *pkt_length,
        char                      *pkt_sechdr_id,
        PGSt_scTime               *pkt_time[8],
        PGSt_double               *secTAI93,
        char                      *pkt_ql_user,
        char                      *pkt_data);
```

FORTRAN:

```
integer function pgs_io_l0_read_pkt (file_handle, spacecrafttag, pkt_idflags,
pkt_apid, pkt_seqflg, pkt_seqcount, pkt_length, pkt_sechdr_id, pkt_time,
sectai93, pkt_ql_user, pkt_data)

        integer             file_handle
        integer             spacecrafttag,
        character*5         pkt_idflags
        integer             pkt_apid
        character*2         pkt_seqflg
        integer             pkt_seqcount
        integer             pkt_length
        character*1         pkt_sechdr_id
        character*8         pkt_time
        double precision    sectai93
        character*8         pkt_ql_user
        character*(TBD)     pkt_data
```

DESCRIPTION: This tool may be used to read a single EDOS or SDPF-generated CCSDS-formatted Level 0 packet.

INPUTS:

file_handle - The file handle returned from the science software for this Level 0 data set file.

spacecraftTag - An identification of the spacecraft platform (i.e., TRMM, AM or PM) from which this telemetry data originates.

OUTPUTS:

pkt_idflags - This field is a combination of the first three fields in the packet primary header and is returned within *pkt_idflags* in the same order. These fields are bit flags which will have a constant value for TRMM, AM and PM platform data. The three fields are the Version Number ("000"), the Type ("0") and Secondary Header Flag ("1").

pkt_apid - The value containing the packet's Application Process ID field from the CCSDS-packet Primary Header 11-bit value.

pkt_seqflg - The value returned is the 2-bit value of the Sequence Flags field found within the packet primary header.

pkt_seqcount - The value containing the packet's Packet Sequence Count field, from the CCSDS-packet Primary Header; a 14-bit field.

pkt_length - The value containing the packet's Packet Length field, from the CCSDS-packet Primary Header 16-bit field.

pkt_sechdr_id - The value returned is the 1-bit value of the Secondary Header ID Flag found in the secondary header of AM and PM telemetry packets. TRMM platform telemetry packets will not contain a Secondary Header ID Flag; using this tool to read a TRMM telemetry packet results in a fill value of "0" to always be returned for this field.

pkt_time - A value of the packet's spacecraft-dependent, bit-packed Time Stamp field, from the CCSDS-packet Secondary Header.

secTAI93 - This value is the packet's spacecraft-dependent Time Stamp field converted to TAI, as real continuous seconds since 12AM UTC 01-01-1993.

pkt_ql_user - The value returned is a combination of the Quicklook and User Flags fields for AM and PM platform telemetry, where the bit fields are returned as individual character values. The 1-bit Quicklook flag is returned as the first character of this field. The User Flags comprise the remaining seven characters of this field; because AM platform telemetry contains only 6-bits of User Flags, the 7th character of the User Flags for AM telemetry will be returned as a fill value of "0". TRMM platform telemetry packets will contain neither Quicklook nor User Flags in the secondary headers of telemetry packets. Because of this, using this tool to read a TRMM telemetry packet results in fill values of "00000000" to always be returned for this field.

pkt_data - A buffer containing each packet's variable length application data field. The length and content vary based on source and content of data.

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX
PGSIO_W_PKT_OUT_OF_RANGE
PGSIO_W_EOF
PGSIO_W_EOF_MORE_STGD_DATA
PGSIO_E_L0_REFERENCE_FAILURE

NOTES: The platform dependent time stamp is not unpacked when it is returned as *pkt_time*. To further unpack this time stamp, the science software must use the returned value of *pkt_time* as input to a time conversion routine such as PGS_TD_SCtime_to_UTC. This routine does, however, return to the user the platform dependent time stamp converted to TAI, as real continuous seconds since 12AM UTC 01-01-1993. This time stamp conversion is performed in order to support the internal SDP Toolkit tracking of the packet times in relation to the user defined data range. This converted packet time value (*secTAI93*) is being returned to the science software to avoid potential duplication of effort and processing. Also, the application data field is not unpacked when it is returned as *pkt_data* to the science software.

EXAMPLES: Below is an example, for an EDOS-generated Level 0 PDS file, of how the function PGS_IO_L0_Read_Pkt can be used to read a single packet. It shows how the function is used in the context of the other SDP Toolkit Level 0 access routines. The example also provides one possible way of reading a sequence of packets based on a time range set by the science software.

C:

```
PGSt_PC_Logical   prodID;     /* file_logical for EDOS PDS file */
PGSt_integer      version;    /* internal counter for multiple physical
                                 files*/
PGSt_IO_Gen_FileHandle  *file_handle;     /* pointer to file handle */
PGSt_SMF_Status   returnStatus;     /* SDP Toolkit status message */
char              edos_hdr_qual[PGSd_IO_MAX_EDOSHDR]; /* buffer
                                 containing EDOS Header and Quality data */
PGSt_integer      num_files;  /* variable for # of staged EDOS PDS files */
PGSt_double       time1_req;  /* variable for start time of time range
                                 requested through ECS planner/scheduler */
PGSt_double       time2_req;  /* variable for end time of time range
```

```
                              requested through ECS planner/scheduler */
PGSt_double       time_first; /* variable for start time of actual time range
                              of staged data*/
PGSt_double       time_last;  /* variable for end time of actual time range of
                              staged data*/
PGSt_double       time_start; /* user defined variable for start time of
                              range of data to be processed */
PGSt_double       time_end;   /* user defined variable for end time of
                              range of data to be processed */
char              fileAttr[PGSd_MAX_FILEATTR];  /* buffer containing
                              file attributes */
PGSt_double       secTAI93;   /* seconds since 12AM UTC 01-01-1993 */
PGSt_integer      pkt_seqcount;     /* variable for packet sequence count */
char              pkt_data[PGSd_IO_MAX_L0_PKT]; /* packet application
                              data field */

/* initialize the user-defined logical identifier for EDOS PDS file */

prodID = 201;

/* call Level 0 processing initialization routine which loads various global
   variables; PGS_IO_L0_Init also returns the number of data files and the
   time range of the staged data */

returnStatus = PGS_IO_L0_Init (prodID, &num_files, &time1_req, &time2_req,
                  &time_first, &time_last, &time_start, &time_end, TBD );

if (returnStatus != PGS_S_SUCCESS)
{
      return (returnStatus);
}
else
{

      /* if necessary, re-set the start and end times for the Level 0 packet
         reads to follow; these values can be set by the user based
         on values returned from the PGS_IO_L0_Init routine for the actual
         start and end times of the staged data (i.e.time_first and time_last,
         which are the default values of time_start and time_end)  */

      time_start = ****;
      time_end = ****;
}

/* loop through and process in turn each physical Level 0 data file */

for (version = 1; version <= num_files; version++)
{
      /* if necessary, read in file attributes for the staged physical file
         corresponding to this version's EDOS PDS file; because the version
         numbers in the PC table may not be time ordered, the Level 0 wrapper
         of PGS_PC_GetFileAttr routine gives the science software
         transparent access to time ordered file attributes */

      returnStatus = PGS_IO_L0_GetFileAttr (prodID, version, fileAttr);

            .
            .
            .
```

```
/* for each desired EDOS PDS file, call a routine to open the EDOS-
   generated file, return the file handle of the EDOS PDS file and also
   return header and quality information located in the EDOS PDS */

returnStatus = PGS_IO_L0_EDOS_Open (prodID, version, edos_hdr_qual,
                  &file_handle);

      .
      .
      .

/* read in packets sequentially while packet time stamp is within
   desired time range */

do
{
      /* read a single packet from the current file position */

      returnStatus = PGS_IO_L0_Read_Pkt (file_handle, spacecraftTag,
            pkt_idflags, &pkt_apid, pkt_seqflg, &pkt_seqcount,
            &pkt_length, pkt_sechdr_id, pkt_time, &secTAI93,
            pkt_ql_user, pkt_data);

      /* test the status returned from the packet read operation */

      switch (returnStatus)
      {
         case PGS_S_SUCCESS:

            /* check if packet number pkt_seqcount corresponds to
               previously quality flagged data */
            /* store packet primary and secondary header
               parameters returned */
            /* store packet application data field pkt_data */
                  .
                  .
                  .
            break;

         case PGSIO_W_PKT_OUT_OF_RANGE:

            /* packet is outside the data time range requested by
               science software */
                  .
                  .
                  .
            break;

         case PGSIO_W_EOF:

            /* at end of file and at end of actual time range of staged
               Level 0 data */
                  .
                  .
                  .
            break;

         case PGSIO_W_EOF_MORE_STGD_DATA:

            /* at end of file for this physical file, more staged
               Level 0 data remains */
```

```
                        .
                        .
                        .
                    break;

                default:
                        .
                        .
                        .
                    break;

            } /* end of switch */

            .
            .
            .

      } while (secTAI93 < time_end); /* end of packet read loop for current
                                        EDOS PDS file */

} /* end of processing loop for staged Level 0 data */
```

FORTRAN:

```
      integer           prodid
      integer           version
      integer           returnstatus
      integer           num_files
      character*(TBD)   fileattr
      integer           pgs_io_l0_init
      integer           pgs_io_l0_getfileattr
      integer           pgs_io_l0_edos_open
      integer           pgs_io_l0_read_pkt
      double precision  time1_req
      double precision  time2_req
      double precision  time_first
      double precision  time_last
      double precision  time_start
      double precision  time_end
      double precision  sectai93
      integer           pkt_seqcount
      character*(TBD)   pkt_data

C     Initialize the user-defined logical identifier for the EDOS PDS file

      prodid = 201

C     Initialize the value of 'sectai93' to the value of 'time_start' in order
C     for the processing loop at statement 400 to begin the first time through

      sectai93 = time_start

C     Call Level 0 processing initialization routine which loads various
C     global variables; pgs_io_l0_init also returns the number of data files
C     and the time range of the staged data

      returnstatus = pgs_io_l0_init(prodid, num_files, time1_req,
     *                  time2_req, time_first, time_last, time_start,
     *                  time_end, TBD )
```

```
      if (returnstatus .ne. pgs_s_success) then

            go to 5000

      else

C           If necessary, re-set the start and end times for the Level 0
C           packet reads to follow; these values can be set by the user
C           based on values returned from the pgs_io_l0_init routine for the
C           actual start and end times of the staged data (i.e. time_first
C           and time_last, which are the default values of time_start and
C           time_end)

            time_start = ****
            time_end = ****
            .
            .
            .

      end if

      .
      .
      .

C     Loop through and process in turn each physical Level 0 data file

      do 300 version = 1, num_files

C           If necessary, read in file attributes for the staged physical file
C           corresponding to this version's EDOS PDS file; because the
C           version numbers in the PC table may not be time ordered, the
C           Level 0 wrapper of pgs_pc_getfileattr routine gives the science
C           software transparent access to time ordered file attributes

            returnstatus = pgs_io_l0_getfileattr(prodid, version, fileattr)

            .
            .
            .

C           For each desired EDOS PDS file, call a routine to open the EDOS-
C           generated files, return the file handle of the EDOS PDS file and
C           also return header and quality information located in the EDOS PDS

            returnstatus = pgs_io_l0_edos_open (prodid, version,
     *                        edos_hdr_qual, file_handle)

            .
            .
            .

C           While the packet times are within the desired time range, read in
C           Level 0 data packets.

  400       if (sectai93 < time_end) then

C                 Read a single packet from the current file position

                  returnstatus = pgs_io_l0_read_pkt (file_handle,
     *                  spacecrafttag, pkt_idflags, pkt_apid, pkt_seqflg,
     *                    pkt_seqcount, pkt_length, pkt_sechdr_id, pkt_time,
     *                    sectai93, pkt_ql_user, pkt_data)

C                   Test the status returned from the packet read operation.

                  if (returnstatus .eq. pgs_s_success) then

C                         Packet was successfully read in, unpack it and store
C                         packet data returned.

                        < check if packet number pkt_seqcount corresponds to
                          previously quality flagged data >
                        < store packet primary and secondary header
                          parameters returned >
                        < store packet application data field pkt_data >
                        .
                        .
                        .

                  elseif (returnstatus .eq. pgsio_w_pkt_out_of_range) then

C                         Packet is outside the data time range requested by
C                         science software
                        .
                        .
                        .
                        go to 450

                  elseif (returnstatus .eq. pgsio_w_eof) then

C                         At end of file and at end of actual time range of
C                         staged Level 0 data
                        .
                        .
                        .
                        go to 450

                  elseif (returnstatus .eq. pgsio_w_eof_more_stgd_data) then

C                         At end of file for this physical file, more staged
C                         Level 0 data remains */
                        .
                        .
                        .
                        go to 450

                  endif
                  .
                  .
                  .
C                   Go back to statement 400 to read in the next data packet.

                  go to 400

            endif

            .
            .
            .

C             Resume Level 0 processing
```

```
 450        Continue

            .
            .
            .
 300  Continue
            .
            .
            .
 5000 <error handling goes here>
```

DETAILS: This routine will make use of global variables that have been previously set by a call to PGS_IO_L0_Init in order to monitor whether a packet is still within the user defined time range of data to be processed. If a packet is out of this time range, the status PGSIO_W_PKT_OUT_OF_RANGE is returned. When reading a packet and end of file is reached, this routine will return a specific status message if more staged Level 0 data exists (PGSIO_W_EOF_MORE_STGD_DATA); otherwise, the status PGS_W_EOF is returned if both the end of file and end of staged data have been reached.

A.7 PGS_IO_L0_Write_Pkt

Write Level 0 Packet (proposed)

NAME: **PGS_IO_L0_Write_Pkt( )**

SYNOPSIS:

C:

```
#include <PGS_IO.h>
#include <PGS_IO_L0.h>

PGSt_SMF_status
PGS_IO_L0_Write_Pkt(
        PGSt_IO_Gen_FileHandle    file_handle,
        PGSt_scTag                spacecraftTag,
        [additional TBD data selection parameters]
        char                      *pkt_idflags,
        PGSt_integer              *pkt_apid,
        char                      *pkt_seqflg,
        PGSt_integer              *pkt_seqcount,
        PGSt_integer              *pkt_length,
        char                      *pkt_sechdr_id,
        PGSt_double               *secTAI93,
        PGSt_scTime               *pkt_time[8],
        char                      *pkt_ql_user,
        char                      *pkt_data
        char                      *lev0_packet);
```

FORTRAN:

integer function pgs_io_l0_write_pkt (file_handle, spacecrafttag, *[additional TBD data selection parameters]*, pkt_idflags, pkt_apid, pkt_seqflg, pkt_seqcount, pkt_length, pkt_sechdr_id, secTAI93, pkt_time, pkt_ql_user, pkt_data, lev0_packet)

```
        integer              file_handle
        integer              spacecrafttag,
        [additional TBD data selection parameters]
        character*5          pkt_idflags
        integer              pkt_apid
        character*2          pkt_seqflg
        integer              pkt_seqcount
        integer              pkt_length
        character*1          pkt_sechdr_id
        double precision     secTAI93
```

character*8	pkt_time
character*8	pkt_ql_user
character**(TBD)*	pkt_data
character**(TBD)*	lev0_packet

DESCRIPTION: This proposed tool is intended for use in science software development and testing, but not for production purposes. This tool may be used to generate and write out a CCSDS-formatted Level 0 packet. Given some TBD input values, such as packet number, time stamp, APID or application data field, this tool will write out a CCSDS-formatted Level 0 packet. Although this tool is expected to create the same form of data packet as are read in by the tool PGS_IO_L0_Read_Pkt, more detailed information and specific calling sequences for this Level 0 access tool will be supplied at a later date, after further design and development.

INPUTS: file_handle - The file handle returned from the science software.

spacecraftTag - The spacecraft identifier desired for the output data.

secTAI93 - The TAI time, as real continuous seconds since 12AM UTC 01-01-1993, that will be converted into the platform-dependent time stamp.

pkt_apid - Application process identifier (APID) of the packet to be generated.

pkt_seqcount - Packet sequence count for the packet to be generated.

pkt_length - Length, in bytes, of the packet to be generated.

pkt_data - Optional user-defined input of the packet application data field.

TBD additional inputs

OUTPUTS: lev0_packet - A packet of Level 0 data. Contains the packet primary and secondary fields and the application data field.

TBD additional outputs

RETURNS: PGSt_SMF_status:

PGS_S_SUCCESS
PGSIO_E_UNIX

NOTES: This tool will simulate the structure of a CCSDS-format Level 0 packet, but will not simulate the science data itself. It is anticipated that the packet returned will contain standard CCSDS-format Level 0 primary and secondary header parameters; a TBD number of these parameters will be simulated and a TBD number will be set to zero or a fill value. The Instrument Data Field in each packet returned will be filled with zeros, unless a user-defined application data field is input to the routine.

This page intentionally left blank.

Endnotes

1. Detailed Mission Requirements (DMR) for the Tropical Rainfall Measuring Mission (TRMM), Issue 2, 513-4DRD/0193 TRMM-500-135, July 1993.

2. Interface Control Document between the Sensor Data Processing Facility (SDPF) and <Mission>, 560-1ICD/0393, August, 1993.

3. Interface Requirements Document Between EOSDIS Core System (ECS) and Tropical Rainfall Measuring Mission (TRMM) Ground System, 194-219-SE1-018, June 1994.

4. Phone conversation with Mike Linda, TSDIS, July 1, 1994.

5. Minutes of meeting between ECS and SDPF, January, 1994.

6. Interface Control Document between the Sensor Data Processing Facility (SDPF) and the X-Ray Timing Explorer (XTE) Customers, 560-1ICD/0993, November, 1993.

7. ECS Product Baseline, May 16, 1994.

8. EOS AM-1 Detailed Mission Requirements (DMR), July 11, 1994.

9. Minutes of May 5, 1994 EDOS-ECS Interface Meeting, Lee Smith.

10. Earth Observing System (EOS) Data and Operations System (EDOS) Functional and Performance Specification, 560-EDOS-0202.0004, November 23, 1992.

11. Earth Observing System (EOS) Data and Operations System (EDOS) Data Format Requirements Document (DFRD), 560-EDOS-0230.0001, December 18, 1992.

12. Tropical Rainfall Measuring Mission (TRMM) System Specification Space Segment, Revision A, TRMM-490-002, July 16, 1993.

13. Tropical Rainfall Measuring Mission (TRMM) Telemetry and Command Handbook, TRMM-490-137, February 21, 1994.

14. Interface Control Document (ICD) Data Format Control Book for EOS-AM Spacecraft (ICD-106), Martin Marietta IS20008658A, April 19, 1994.

15. General Instrument Interface Specification, EOS-AM Project, GSFC-420-03-02, December 1, 1992.

16. General Interface Requirements Document (GIRD) for EOS Common Spacecraft/ Instruments, EOS PM Project, Revision A, GSFC 422-11-12-01, January 1994.

17. TRMM Spacecraft Housekeeping Telemetry Packets, Bruce Love, TRMM Project, May 11, 1994.

18. Tropical Rainfall Measuring Mission (TRMM) Observatory to Space Flight Tracking and Data Network (STDN) Radio Frequency (RF) Interface Control Document, TRMM-490-086, April 15, 1993.

19. Tropical Rainfall Measuring Mission (TRMM) Spacecraft to Clouds and the Earth's Radiant Energy System (CERES) Instrument ICD, TRMM-490-021, April 30, 1993.

20. Tropical Rainfall Measuring Mission (TRMM) Spacecraft to Lightning Imaging Sensor (LIS) Instrument ICD, TRMM-490-022, February 26, 1993.

21. Phone conversation with Bruce Love, TRMM Project, July 29, 1994.

22. Minutes of FDF-ECS/PGS Interface Meeting, August 5, 1993.

23. PGS Toolkit User's Guide for the ECS Project, Version 1, Final, 194-809-SD4-001, May 1994.

24. Draft CERES Data Catalog Product List, CERES Instrument Team, September 10, 1993.

25. CERES Instrument Operations Overview presentation vugraphs, EOS-AM Operations Workshop, Larry Blumfield, April 22, 1993.

26. Phone conversation with Bill Weaver, CERES, July 5, 1994.

27. Phone conversation with Bruce Love, TRMM Project, January 21, 1994.

28. Presentation vugraphs for EOS-AM Spacecraft Program Communication/Command & Data Handling (COMM/C&DH) Subsytem Housekeeping Overview, Martin Marietta, April 14-15, 1994.

29. Overview of Interfaces Between the Flight Dynamics Facility (FDF) and the EOSDIS Core System (ECS), 554-FDD-91/084 CSC/TM-91/6068, June 1991.

30. Flight Dynamics Division (FDD) Generic Data Product Formats Interface Control Document, 533-FDD-91/028, June 1991.

31. Minutes of FDF-ECS Interface Meeting, December 20, 1993.

32. NASA/SPSO Data Granule Survey, April 12, 1994.

33. Level 0 Simulation Discussion presentation vugraphs, Dan Marinelli, DPFT V meeting, April 7, 1994.

34. Tropical Rainfall Measuring Mission (TRMM) Test and Training Simulator (TTTS) Functional Requirements Document (FRD), 515-4FRD/0194, TRMM-500-154, February 1994.

35. EOSDIS Test System (ETS) Operations Concept, System Requirements Review (SRR), Version 2, 515-ETS-01-V2, December 1993.

36. Phone conversation with Don Jamison, NASA/EOSDIS Data Transport Manager, January 7, 1994.

37. Conversation with George Turner, NASA/Code 563.2, June 1994.

Abbreviations and Acronyms

ACS	Attitude Control System
AMSU	Advanced Microwave Sounding Unit
API	Application Programming Interface
APID	Application Process Identifier
ASTER	Advanced Spaceborne Thermal Emission and Reflection Radiometer
C&DH	Command and Data Handling
CCR	Configuration Change Request
CCSDS	Consultative Committee for Space Data Systems
CDS	CCSDS Day Segmented
CERES	Clouds and Earth's Radiant Energy System
CRC	Cyclic Redundancy Code
CUC	Constant and Unit Conversions
DAAC	Distributed Active Archive Center
DADS	Data Archive and Distribution System
DAN	Data Availability Notice
DAS	Data Availability Schedule
DCF	Data Capture Facility
DDF	Data Distribution Facility
DFRD	Data Formats Requirements Document
DIF	Data Interface Facility
DMR	Detailed Mission Requirements
DPF	Data Processing Facility
DPFT	Data Processing Focus Team
ECI	Earth Centered Inertial
ECOM	EOS Communications
ECS	EOSDIS Core System
EDOS	EOS Data and Operations System
EGS	EOS Ground System

EOC	EOS Operations Center
EOS	Earth Observing System
EOSDIS	EOS Data and Information System
EOSP	Earth Observing Scanning Polarimeter
EPDS	Earth Probe Data System
EPV	Extended Precision Vector
ESDIS	Earth Science Data and Information System
ETS	EOSDIS Test System
F&PS	Functional and Performance Specification
FDD	Flight Dynamics Division
FDF	Flight Dynamics Facility
FRD	Functional Requirements Document
GDS	Ground Data System
GIIS	General Instrument Interface Specification
GSFC	Goddard Space Flight Center
GTSIM	Generic Telemetry Simulator
ICD	Interface Control Document
IMS	Information Management System (ECS)
IP	International Partner
IRD	Interface Requirement Document
IRU	Inertial Reference Unit
km	kilometer
LaRC	Langley Research Center
LIS	Lightning Imaging Sensor
MDUE	Missing Data Unit Entry
MDUL	Missing Data Unit List
MHS	Microwave Humidity Sounder
MIMR	Multi-Frequency Imaging Microwave Radiometer
MISR	Multi-Angle Imaging Spectro-Radiometer
MOC	Mission Operations Center
MODIS	Moderate Resolution Imaging Spectrometer

MOPITT	Measurement of Pollution in the Troposphere
MSFC	Marshall Space Flight Center
NASA	National Aeronautics and Space Administration
NASCOM	NASA Communications Network
NASDA	National Space Development Network (Japan)
O/A	Orbit/Attitude
ODC	Other Data Center
Pacor	Packet Processor
PB	Petabyte
PDR	Performance Design Review
PDS	Production Data Set
PGE	Product Generation Executable
PGS	Product Generation System
QA	Quality Assurance
QAC	Quality Accounting Capsule
RF	Radio Frequency
SAA	South Atlantic Anomaly
SCF	Science Computing Facility
SDP	Science Data Processing
SDPF	Sensor Data Processing Facility
SDPS	Science Data Processing Segment
SDR	System Design Review
SFDU	Standard Formatted Data Unit
SMC	System Management and Coordination
SOHO	Solar and Heliospheric Observatory
SPSO	EOS Project Science Office
SRR	System Requirements Review
SSIM	Spacecraft Simulator
STDN	Space Tracking and Data Network
SWAMP	Science Working group for the AM Platform
SWAS	Sub-Millimeter Wave Astronomy Satellite

TAI	International Atomic Time
TBC	To Be Confirmed
TBD	To Be Determined
TBS	To Be Supplied
TDB	Barycentric Dynamical Time
TDRS	Tracking and Data Relay Satellite
TDRSS	Tracking and Data Relay Satellite System
TDT	Terrestrial Dynamical Time
TMI	TRMM Microwave Imager Instruments
TONS	TDRSS Onboard Navigation System
TRMM	Tropical Rainfall Measuring Mission
TSDIS	TRMM Science Data and Information System
TSS	TDRSS Support Session
TTS	Test and Training Simulator
UTC	Coordinated Universal Time
UTCF	Universal Time Correlation Factor
VCDU	Virtual Channel Data Unit
VIRS	Visible Infrared Scanner
WSC	White Sands Complex
XTE	X-Ray Timing Explorer